NAVAL HEALTH RESEARCH CENTER

TECHNICAL MANUAL FOR THE NAVY COMPUTER ASSISTED MEDICAL DIAGNOSIS KNOWLEDGE BASE EDITOR (NCAMD-KBE)

Version 1.0.

H. L. Ly

D. M. Pearsall

Technical Document 96-4D

NAVAL HEALTH RESEARCH CENTER
P. O. BOX 85122
SAN DIEGO, CALIFORNIA 92186 – 5122

NAVAL MEDICAL RESEARCH AND DEVELOPMENT COMMAND
BETHESDA, MARYLAND

Technical Manual for the Navy Computer Assisted Medical Diagnosis Knowledge Base Editor (NCAMD-KBE), Version 1.0.

Prepared by:

Hoa L. Ly
Dianna M. Pearsall

Naval Health Research Center
Medical Information Systems and
Operations Research Department
P.O. Box 85122
San Diego, CA 92186-5122

Technical Document 96-4D supported by the Naval Medical Research and Development Command, Bethesda, MD, Department of the Navy, under Work Unit No. 63706N M0095.005-6103. The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Navy, Department of Defense, nor the U.S. Government.

SUMMARY

This technical manual contains the information on the program source code, data elements, and the database structure needed to maintain the Navy Computer Assisted Medical Diagnosis Knowledge Base Editor (NCAMD-KBE). This documentation was created using the FOXDOC Version 2.5a program.

TABLE OF CONTENTS

Introduction

This document is the technical guide for the Knowledge Base Editor within the Navy Computer Assisted Medical Diagnosis (NCAMD) System. FoxDoc Version 2.5a was used to generate the program code in Section VII. This technical manual describes the knowledge base data structures and reviews the diagnostic algorithm and data flows. It contains seven sections:

1. System Summary
2. Menu Summary
3. Screen Summary
4. Data Dictionary
5. Tree Diagram
6. Procedure Summary
7. Source Code Program Listing

Section I. System Summary. See the tree diagram in Section V for programs, procedures, functions, and file formats.

This system has:

```
13520 lines of code
    1 program file
   32 procedure files
  178 procedures and functions
   16 table/dbfs
   17 index files
    1 menu file
   23 screen files
    2 other files
  612 cross-referenced tokens
```

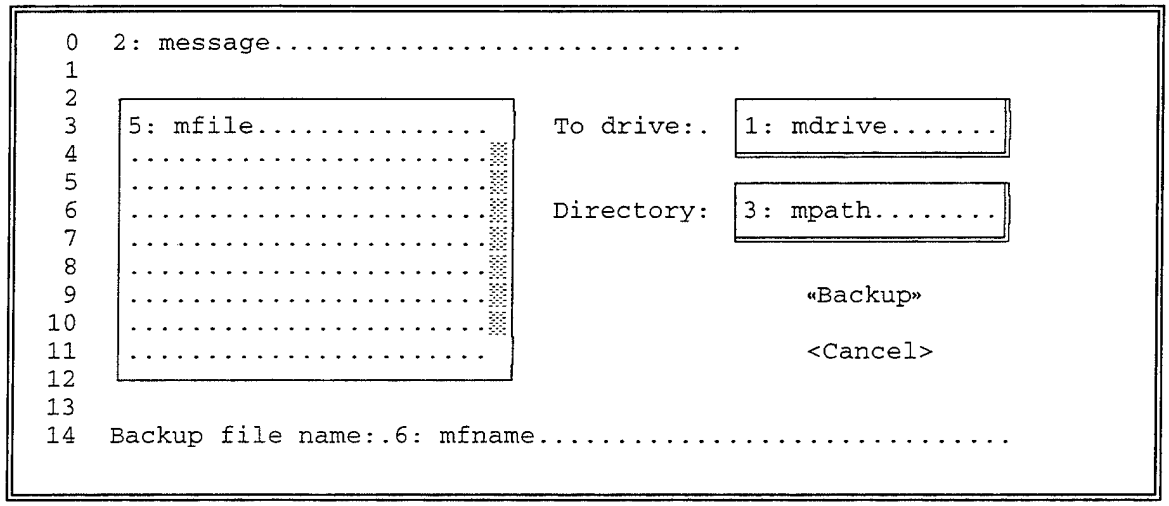
Section II. Menu File Summary. The system has one menu: KBMENU.MNX.

MENU OPTIONS	MENU OPTION IDENTIFIERS
System	
Help F1	System_MST_HELP

Backup	
Restore	
Knowledge Base	KNOWLEDGEB
Knowledge Base Area	
Question - Data	
Dictionary	
Exit system	

Section III. Screen Summary. The system has twenty three (23) screen files: BACKUP, KB, KBDEL, QUAL, GOAL, DISPLAY, DISEASE, ACTION, ACTELSE, CLAUSE, RULE, TERM, OBJECT, ENUM, RESTORE, DD, PLAN, DICT, KBEDIT, OBLIST, DDEDIT, DDENUM, and KBLOAD.

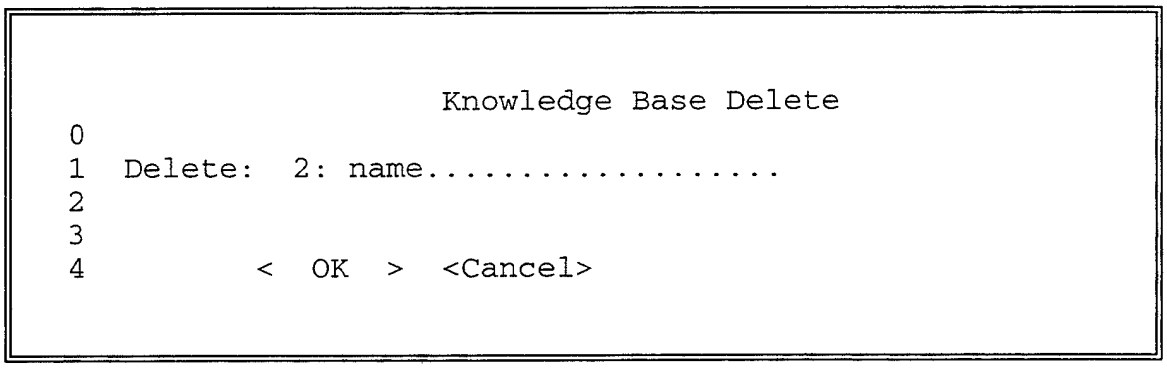
A. BACKUP.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_backup
Coordinates: FROM 0,0 TO 0,60
Window options: FLOAT CLOSE MINIMIZE SHADOW

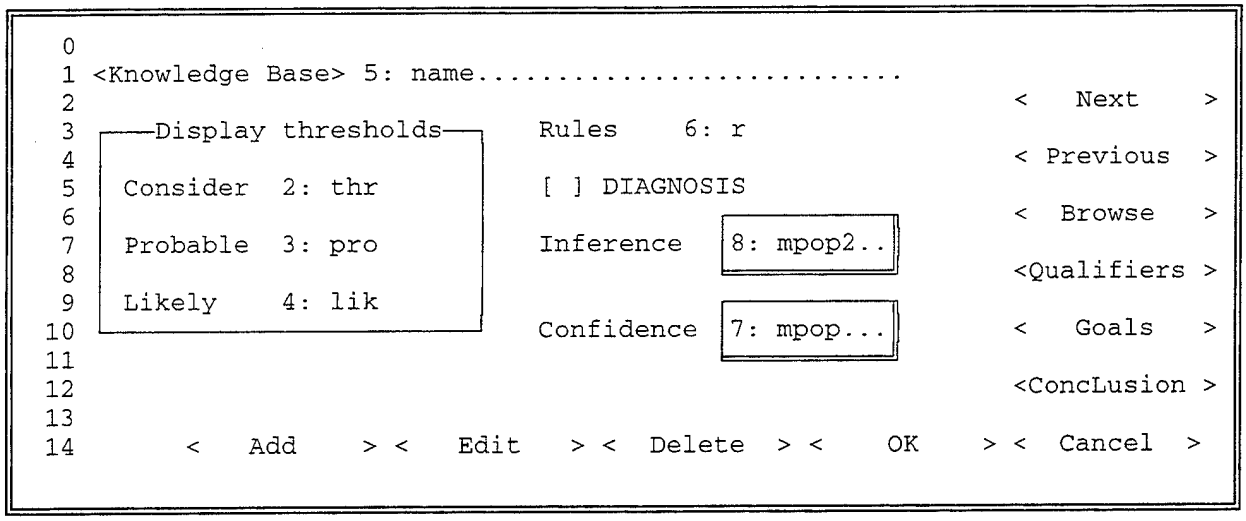
	Name	Type	Picture
1:	mdrive	Popup	"@^ "
2:	message	Field	
3:	mpath	Popup	"@^ "
4:	maction	Push button	"@*HT \!\<Backup;\<Cancel"
5:	mfile	List	"@&N"
6:	mfname	Field	
7:	mdrive	Popup	"@^ "
8:	message	Field	
9:	mpath	Popup	"@^ "
10:	maction	Push button	"@*VT \!\<Backup;\<Cancel"
11:	mfile	List	"@&N"
12:	mfname	Field	

B. KBDEL.SCX **Last updated: 10/03/94 at 15:01**



	Name	Type	Picture
1:	mbutton	Push button	"@*HT OK;Cancel"
2:	area.name	Field	
3:	mbutton	Push button	"@*HT OK;Cancel"
4:	area.name	Field	

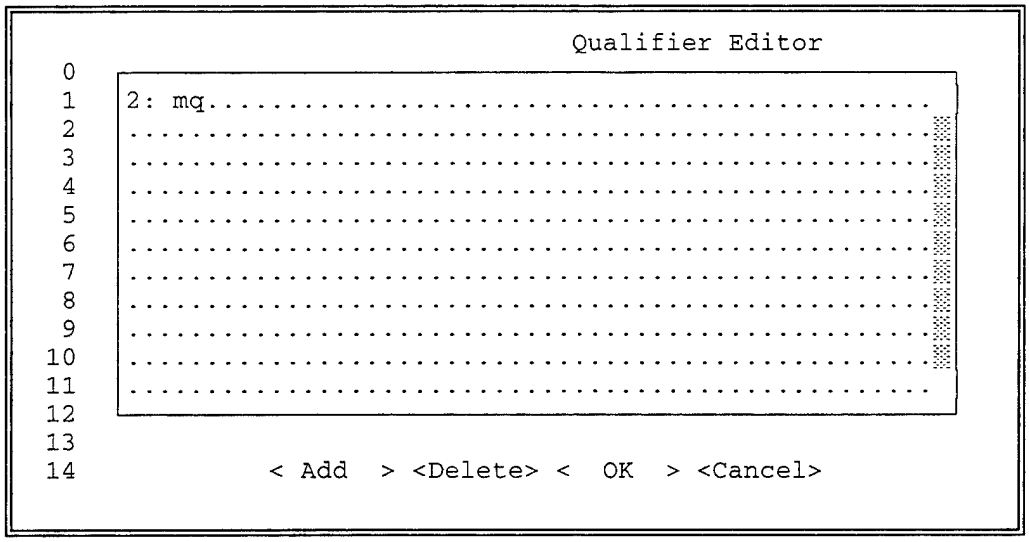
C. KB.SCX Last updated: 10/03/94 at 15:01



Window name: W_kb
Coordinates: FROM 0,0 TO 0,74
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*HN
2:	mbbuttons	Push button	"@*VN
3:	m.areaname	Field	
4:	m.threshold	Field	
5:	m.probable	Field	
6:	m.likely	Field	
7:	m.rules	Field	
8:	m.isdiag	Check box	"@*C DIAGNOSIS"
9:	m.mpop	Popup	"@^ BAYES;ADDITIVE"
10:	m.mpop2	Popup	"@^ FORWARD;BACKWARD"
11:	mbuttons	Push button	"@*HT
12:	m.threshold	Field	
13:	m.probable	Field	
14:	m.likely	Field	
15:	m.name	Field	
16:	m.rules	Field	
17:	m.mpop	Popup	"@^ BAYES;ADDITIVE"
18:	m.mpop2	Popup	"@^ FORWARD;BACKWARD"
19:	m.isdiag	Check box	"@*C DIAGNOSIS"
20:	minvbutton	Push button	"@*HN \<Knowledge Base"
21:	mbbuttons	Push button	"@*VN

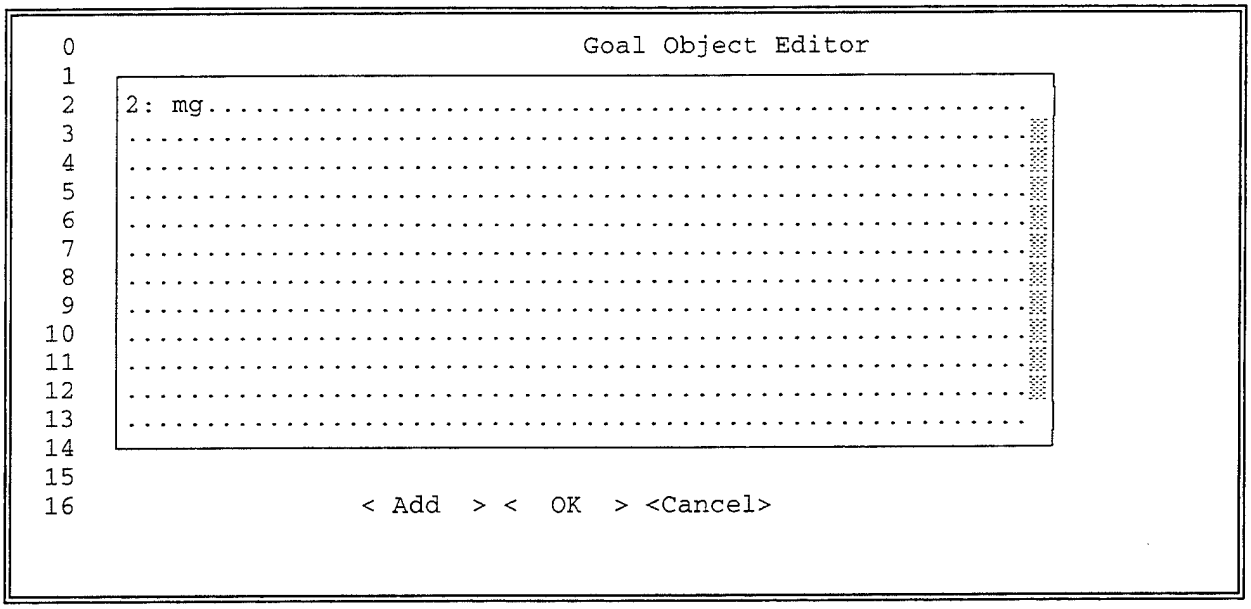
D. QUAL.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_qe
Coordinates: FROM 0,0 TO 12,57
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*HT \<Edit;\<Quit"
2:	mq	List	"@&N"
3:	mbuttons	Push button	"@*HN
4:	mq	List	"@&N"

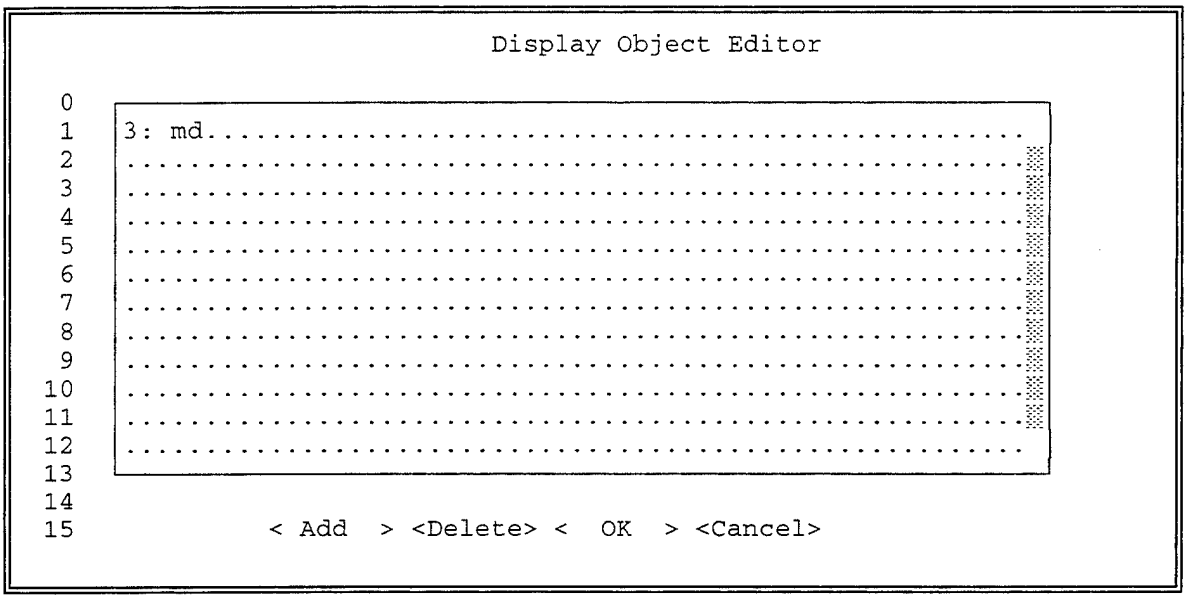
E. GOAL.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_goal
Coordinates: FROM 0,0 TO 13,63
Window options: FLOAT CLOSE MINIMIZE SHADOW

Name	Type	Picture
1: mbuttons	Push button	"@*HN \<Edit;\<Quit"
2: mg	List	"@&N"
3: mbuttons	Push button	"@*HN \<Add;\<OK;\<Cancel"
4: mg	List	"@&N"

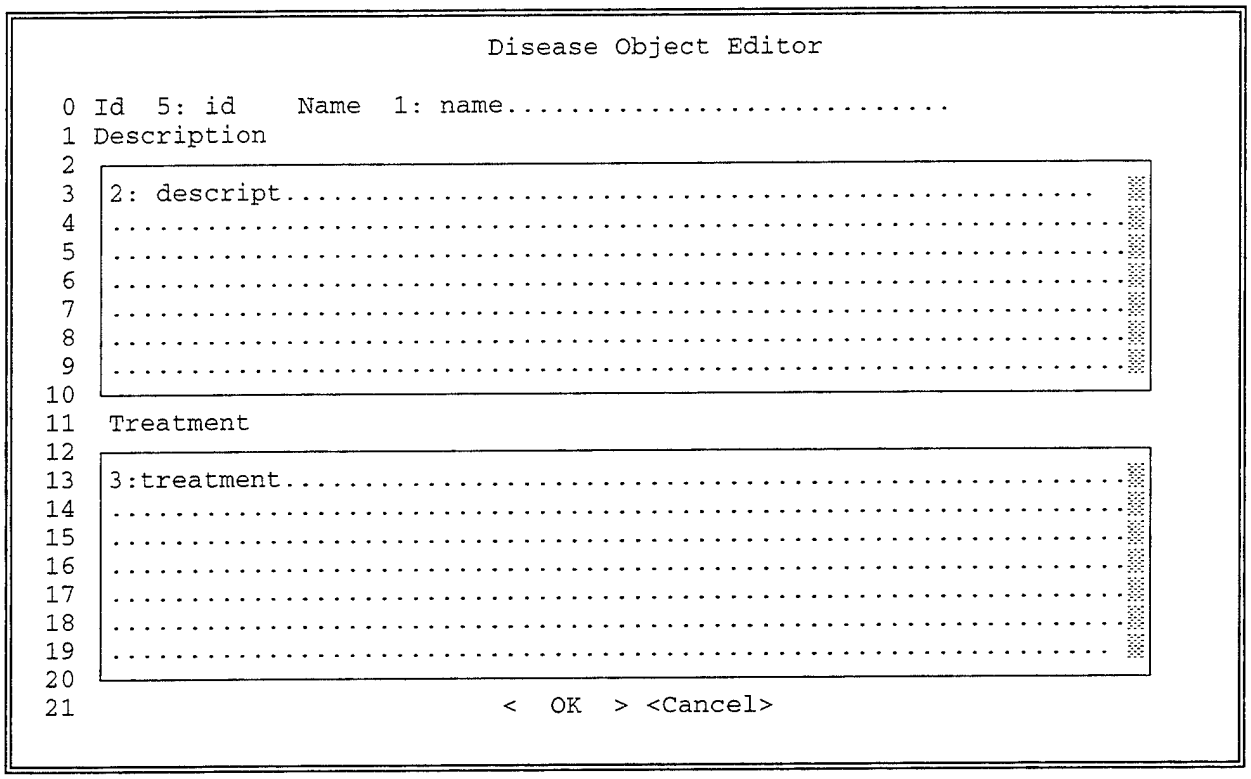
F. DISPLAY.SCX Last updated: 10/03/94 at 15:01



Window name: W_displ
Coordinates: FROM 0,0 TO 13,63
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*HT \<Edit;\<Quit"
2:	md	List	"@&N"
3:	mbuttons	Push button	"@*HT OK;Cancel"
4:	mbbutton	Push button	"@*HN Add;Delete"
5:	md	List	"@&N"

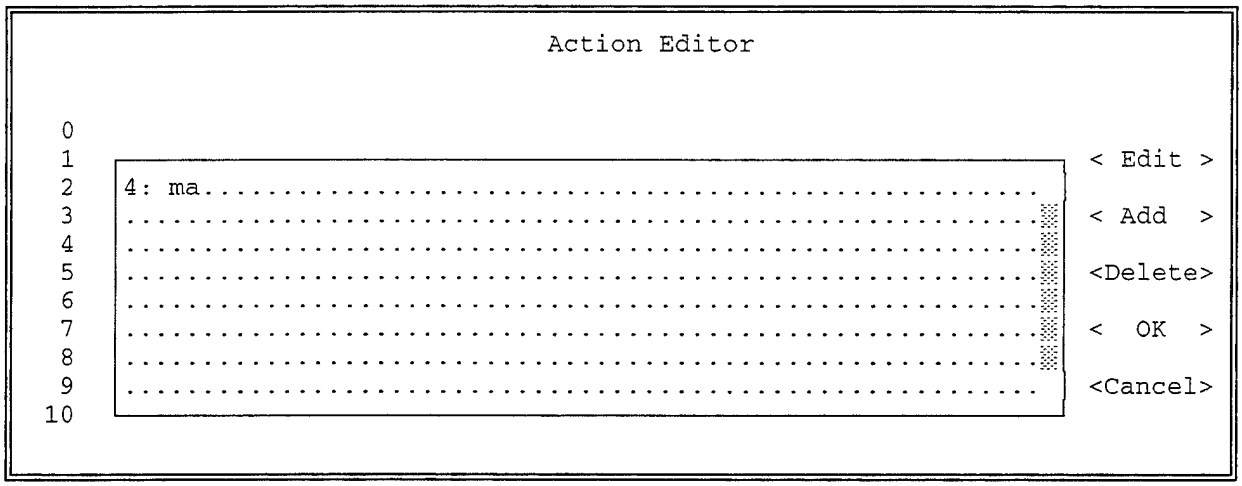
G. DISEASE.SCX Last updated: 10/03/94 at 15:01



Window name: W_disease
Coordinates: FROM 0,0 TO 0,77
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	m.id	Field	
2:	m.name	Field	"@!"
3:	m.descript	Field	
4:	m.treatment	Field	
5:	mbuttons	Push button	"@*HT \<OK;\<Cancel"
6:	disease.name	Field	"@!"
7:	disease.descript	Field	
8:	disease.treatment	Field	
9:	mbuttons	Push button	"@*HN \<OK;\<Cancel"
10:	disease.id	Field	

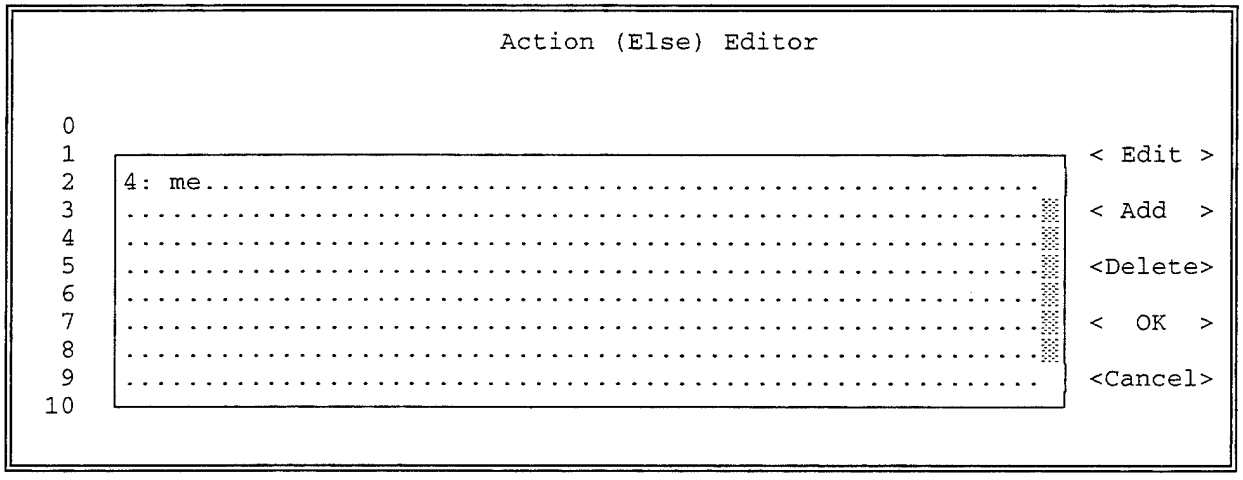
H. ACTION.SCX Last updated: 10/03/94 at 15:01



Window name: W_act
Coordinates: FROM 0,0 TO 9,75
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*VN \<Edit;\<Quit"
2:	ma	List	"@&N"
3:	mbuttons	Push button	"@*VT \<Delete;\<OK>"
4:	mebutton	Push button	"@*VN \<Edit"
5:	minsbutton	Push button	"@*VN \<Add"
6:	ma	List	"@&N"

I. ACTELSE.SCX Last updated: 10/03/94 at 15:01

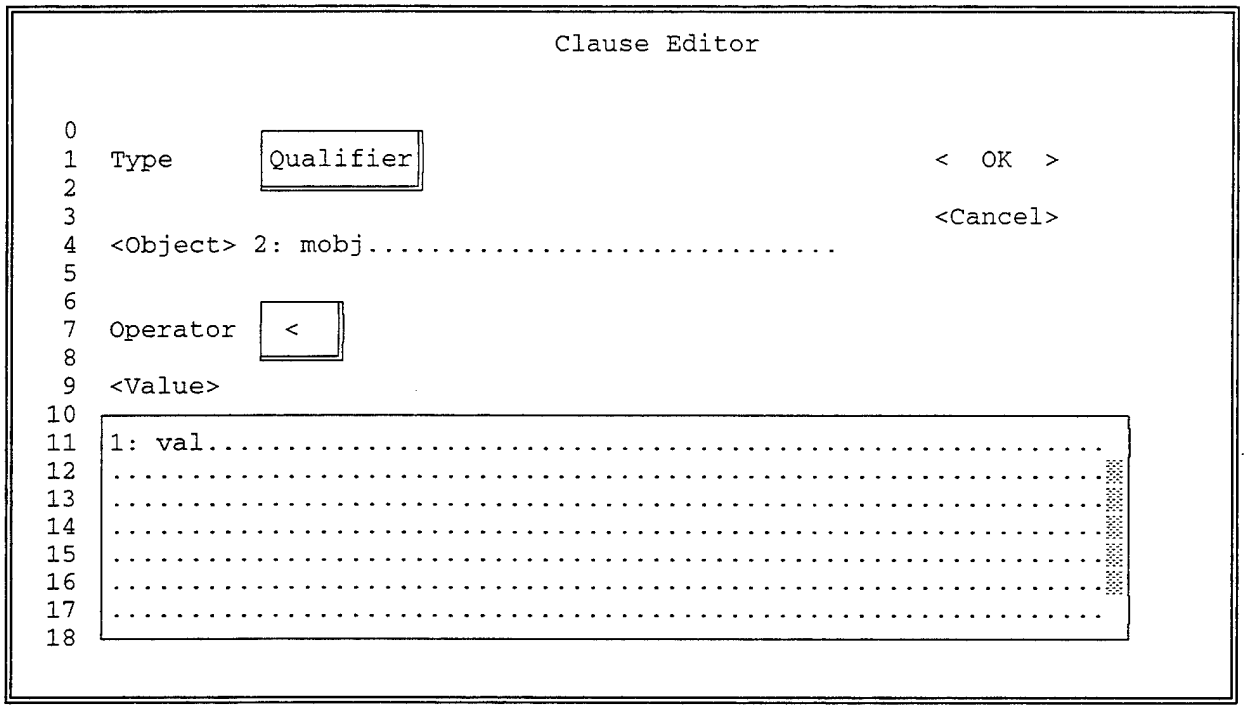


```
Window name: W_act
Coordinates: FROM 0,0 TO 9,75
Window options: FLOAT CLOSE MINIMIZE SHADOW

    Name                Type            Picture

1:  mbuttons            Push button     "@*VN \<Edit;\<Quit"
2:  me                  List            "@&N"
3:  mbuttons            Push button     "@*VT \<Delete;\<OK>"
4:  mebutton            Push button     "@*VN \<Edit"
5:  minsbutton          Push button     "@*VN \<Add"
6:  me                  List            "@&N"
```

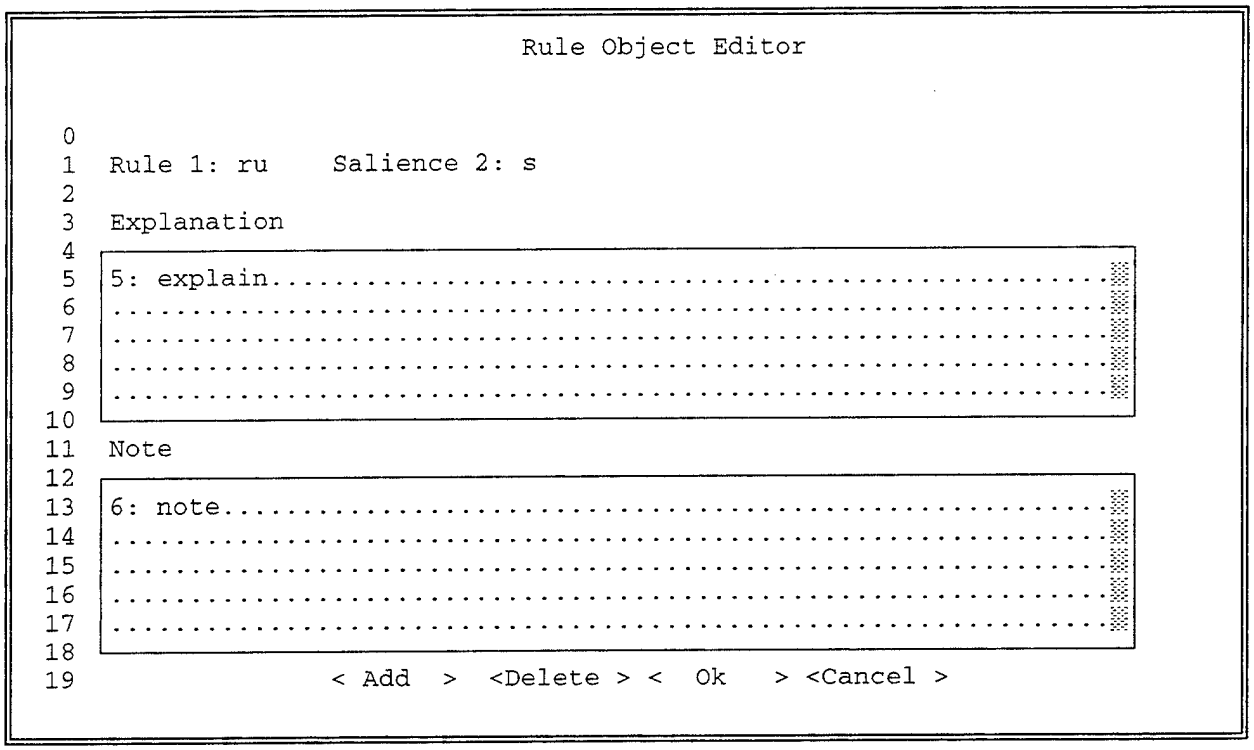
J. CLAUSE.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_clause
Coordinates: FROM 0,0 TO 0,67
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*VT \<OK;\<Cancel"
2:	mtype	Popup	"@^ Qualifier;Goal"
3:	mobj	Field	
4:	m.op	Popup	"@^ <;<=;==;>=;>;!=;="
5:	m.val	Field	
6:	m.val	Field	
7:	mobj	Field	
8:	m.op	Popup	"@^ <;<=;==;>=;>;!=;="
9:	mtype	Popup	"@^ Qualifier;Goal"
10:	mbuttons	Push button	"@*VT \<OK;\<Cancel"

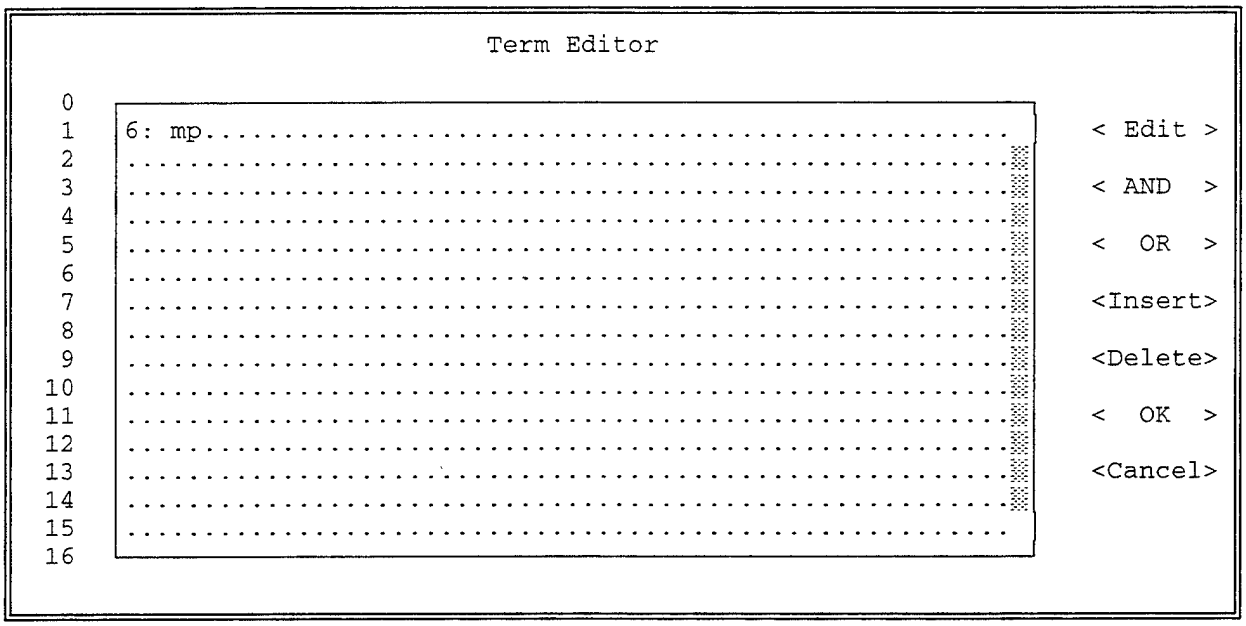
K. RULE.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_rule Window name: W_rule
Coordinates: FROM 0,0 TO 5,75
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	m.rule	Field	
2:	m.salience	Field	
3:	mbuttons	Push button	"@*HT \<OK ;\<Cancel"
4:	m.explain	Field	
5:	m.note	Field	
6:	m.rule	Field	
7:	m.salience	Field	
8:	mbbutton	Push button	"@*VN \<Add"
9:	mbuttons	Push button	"@*HT
10:	m.explain	Field	
11:	m.note	Field	

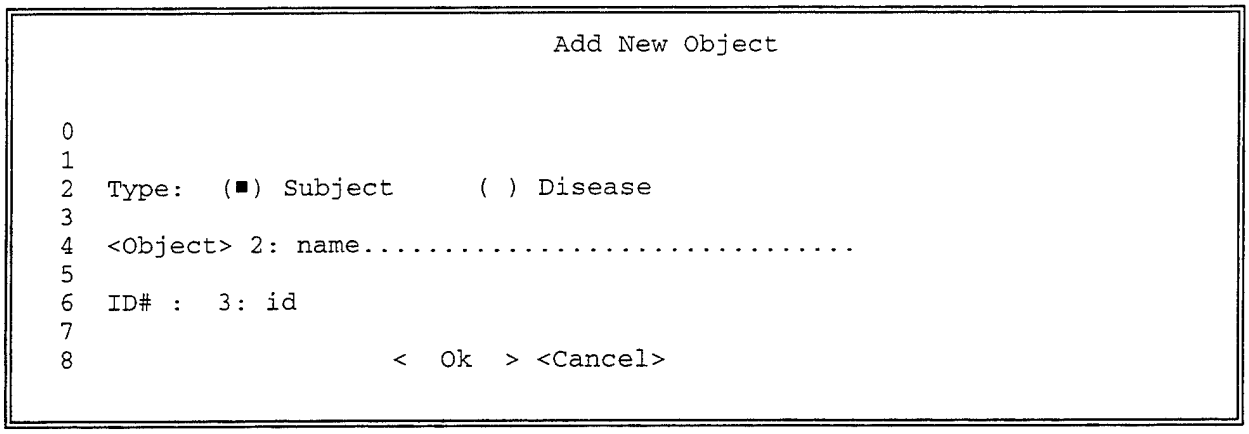
L. TERM.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_term
Coordinates: FROM 0,0 TO 16,75
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mp	List	"@&N"
2:	mbuttons	Push button	"@*VN \<Edit;\<Quit"
3:	mbuttons	Push button	"@*VT \<Del
4:	mbbutton	Push button	"@*VN \<AND"
5:	morbutton	Push button	"@*VN \<OR"
6:	mebutton	Push button	"@*VN \<Edit"
7:	minsbutton	Push button	"@*VN \<Insert"
8:	mp	List	"@&N"

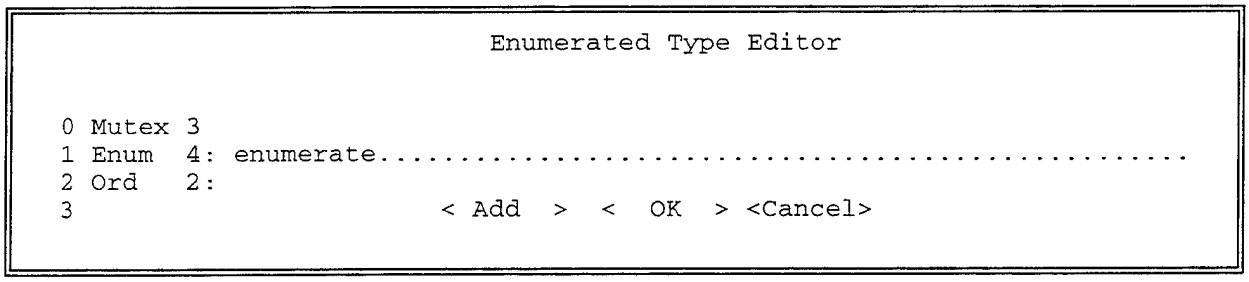
M. OBJECT.SCX **Last updated: 10/03/94 at 15:01**



Window name: Object
Coordinates: FROM 0,0 TO 0,53
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	m.object	Radio button	"@*RHN Subject;Disease"
2:	m.name	Field	
3:	m.id	Field	
4:	m.button	Push button	"@*HT \<OK;\<Cancel"
5:	m.obj	Push button	"@*HN Object"
6:	m.object	Radio button	"@*RHN Subject;Disease"
7:	m.name	Field	
8:	m.id	Field	
9:	m.button	Push button	"@*HT \<OK;\<Cancel"
10:	m.obj	Push button	"@*HN Object"

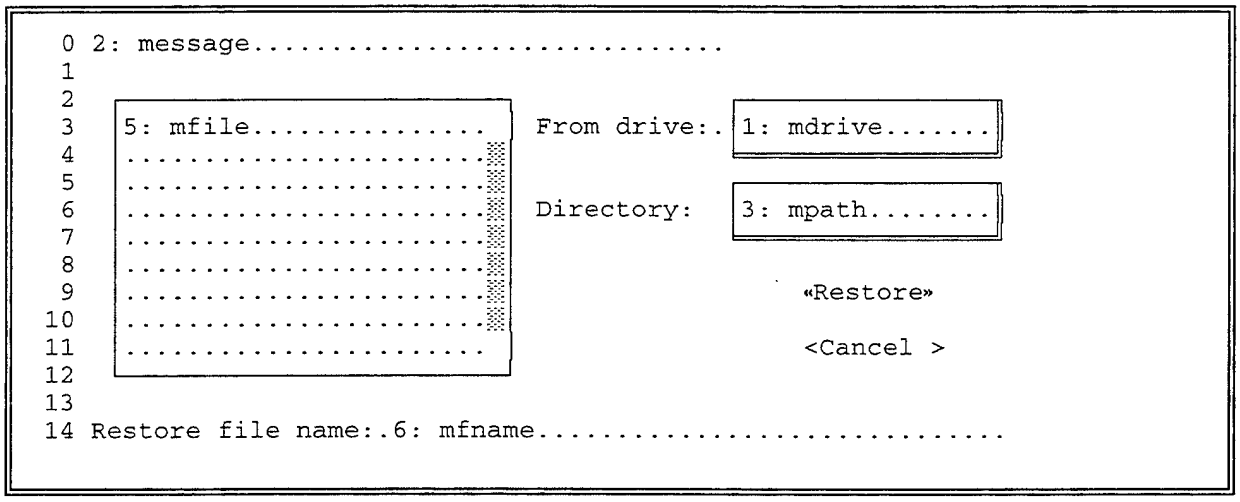
N. ENUM.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_qenum
Coordinates: FROM 0,0 TO 0,70
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mbuttons	Push button	"@*HT OK;Cancel"
2:	enum.ord	Field	
3:	enum.mutex	Field	
4:	enum.enumerate	Field	"@!"
5:	mbbutton	Push button	"@*VN Add"
6:	mbuttons	Push button	"@*HT OK;Cancel"
7:	enum.ord	Field	
8:	enum.mutex	Field	
9:	enum.enumerate	Field	"@!"
10:	mbbutton	Push button	"@*VN Add"

O. RESTORE.SCX **Last updated: 10/03/94 at 15:01**

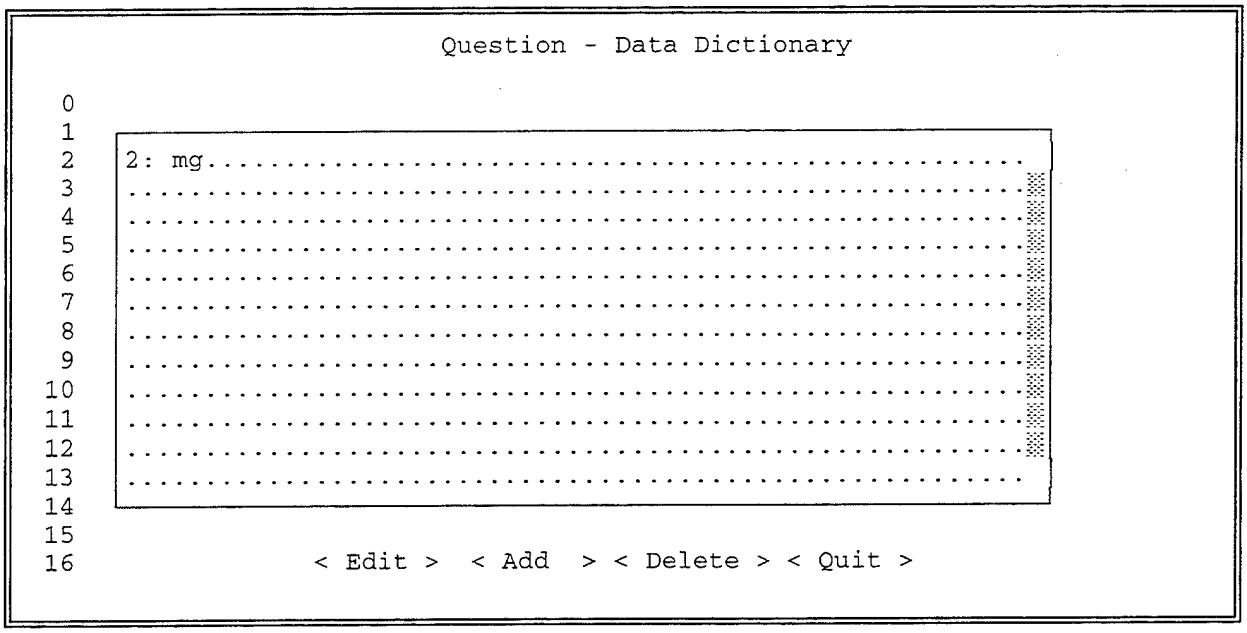


Window name: W_restore
Coordinates: FROM 0,0 TO 0,60
Window options: FLOAT CLOSE MINIMIZE SHADOW

	Name	Type	Picture
1:	mdrive	Popup	"@^ "
2:	message	Field	
3:	mpath	Popup	"@^ "
4:	maction	Push button	"@*HT \!\<Restore;\<Cancel"
5:	mfile	List	"@&N"
6:	mfname	Field	
7:	mdrive	Popup	"@^ "
8:	message	Field	
9:	mpath	Popup	"@^ "
10:	maction	Push button	"@*VT \!\<Restore;\<Cancel"

11: mfile	List	"@&N"
12: mfname	Field	

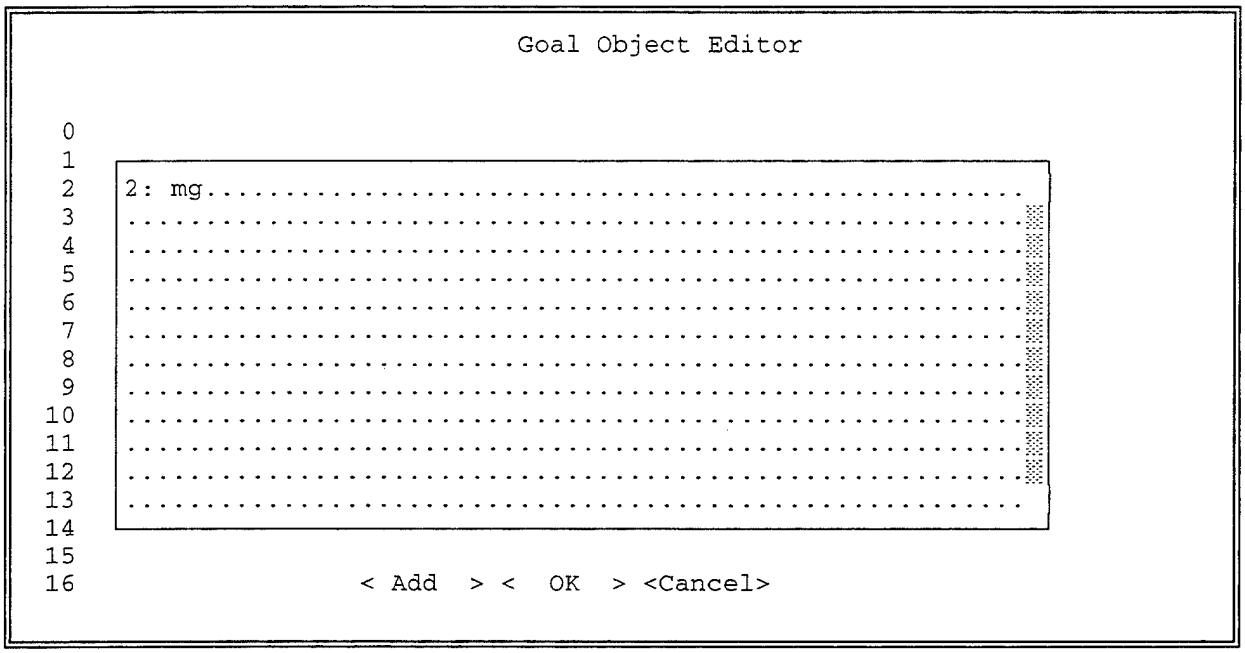
P. DD.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_
Coordinates: FROM 0,0 TO 0,60
Window options: FLOAT CLOSE MINIMIZE SHADOW

Name	Type	Picture
1: mbuttons	Push button	"@*HN \<Edit;\<Add;
2: mq	List	"@&N"

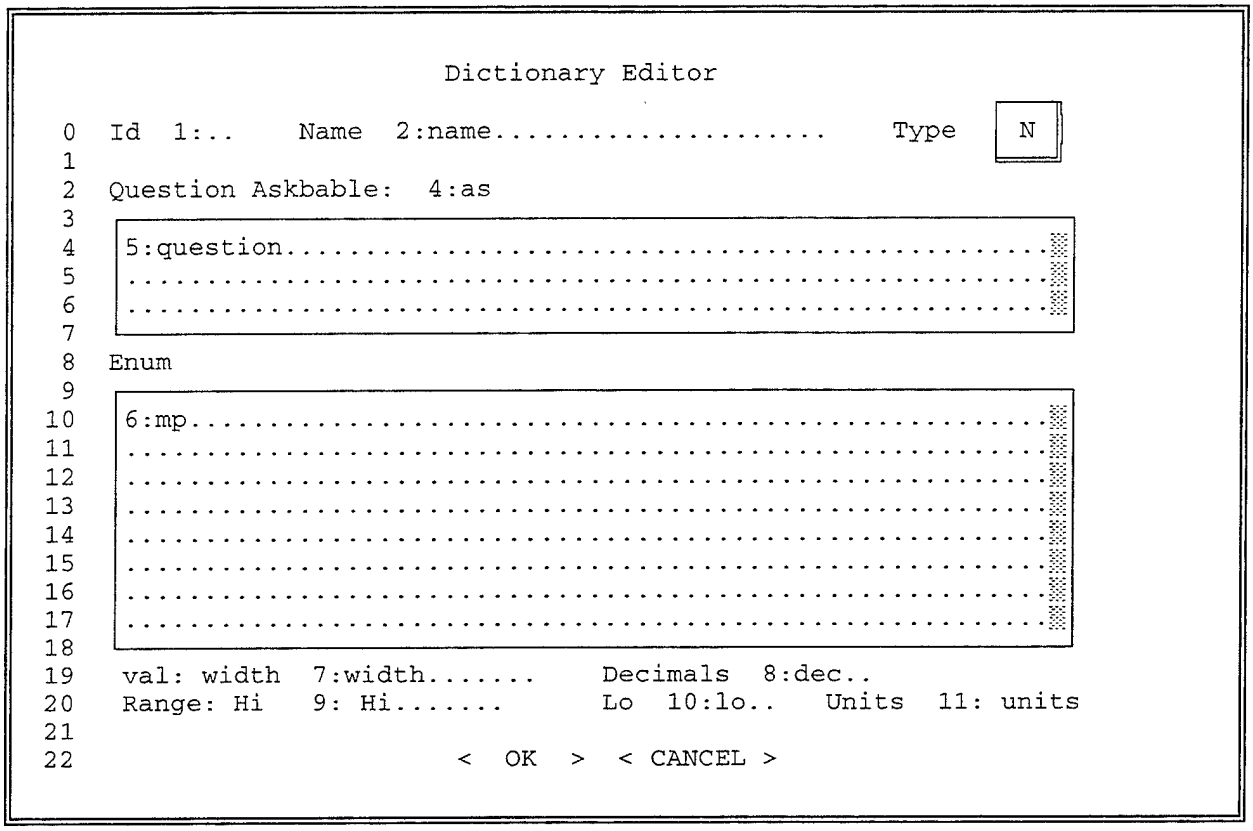
Q. PLAN.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_goal
Coordinates: FROM 0,0 TO 13,63
Window options: FLOAT CLOSE MINIMIZE SHADOW

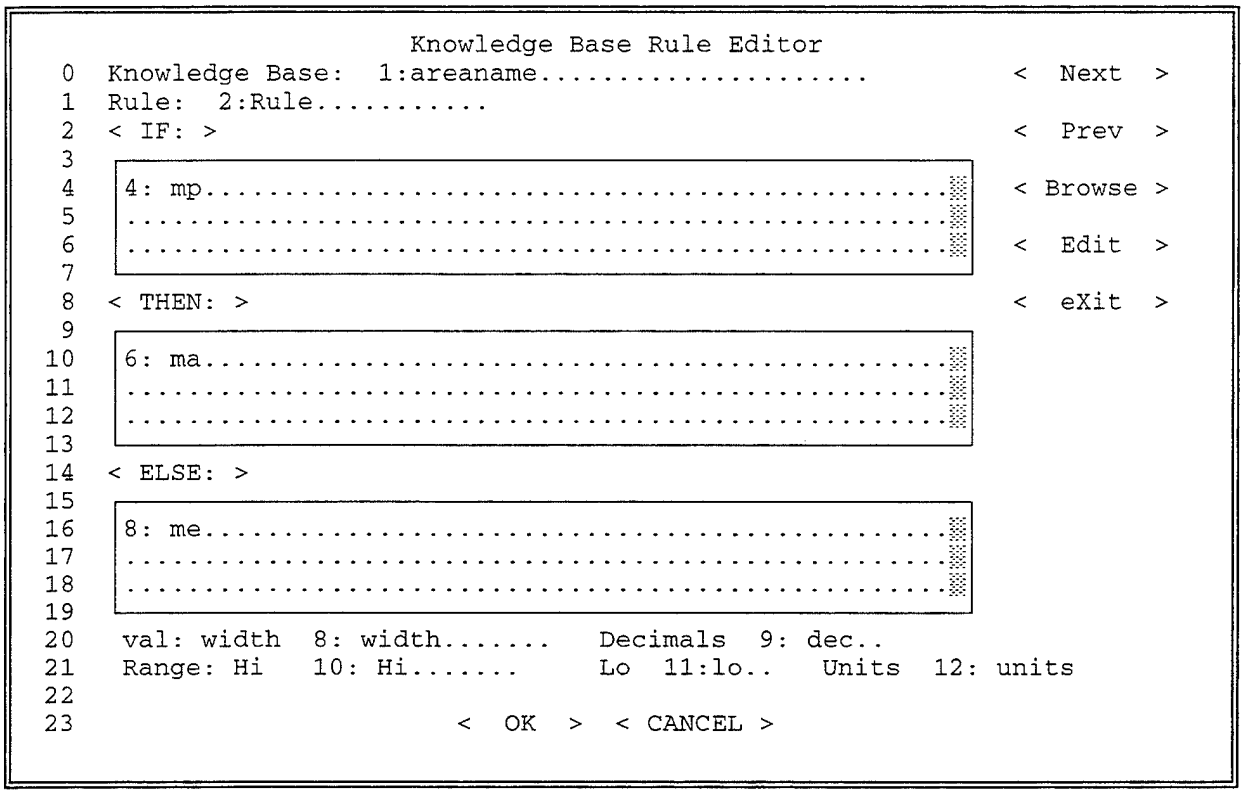
Name	Type	Picture
1: mbuttons	Push button	"@*HN \<Add;\<OK;\<Cancel"
2: mg	List	"@&N"
3: mbuttons	Push button	"@*HN \<Add;\<OK;\<Cancel"
4: mg	List	"@&N"

R. DICT.SCX **Last updated: 10/03/94 at 15:01**



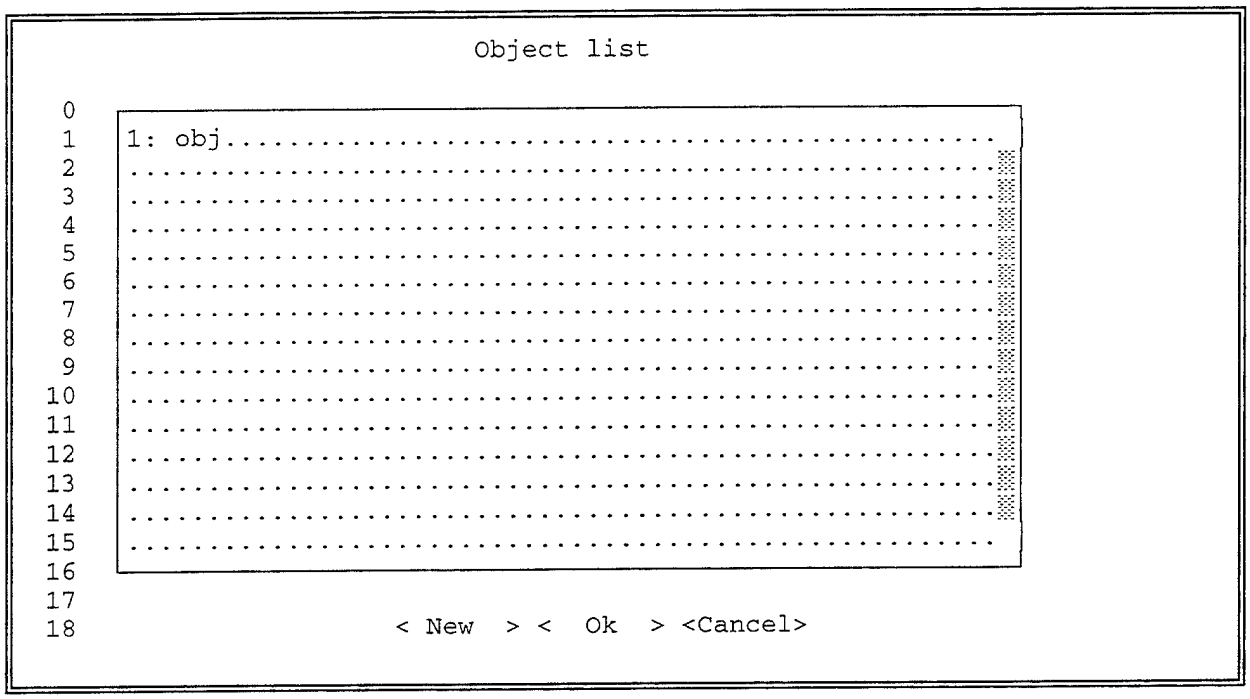
	Name	Type	Picture
1:	m.id	Field	
2:	m.name	Field	
3:	m.datatype	Popup	"@^ N;E;M;L"
4:	dict.askable	Field	
5:	m.question	Field	
6:	mp	List	"@&N"
7:	m.width	Field	
8:	m.dec	Field	
9:	m.hi	Field	
10:	m.lo	Field	
11:	m.units	Field	
12:	mbuttons	Push button	"@*HT OK;Cancel"

S. KBEDIT.SCX Last updated: 10/03/94 at 15:01



Name	Type	Picture
1: m.areaname	Field	
2: rule.rule	Field	
3: minvprem	Push button	"@*HN \<IF:"
4: mp	List	"@&N"
5: minvact	Push button	"@*HN \<THEN:"
6: ma	List	"@&N"
7: minvelse	Push button	"@*HN \<ELSE:"
8: me	List	"@&N"
9: mbbutton	Push button	"@*VN

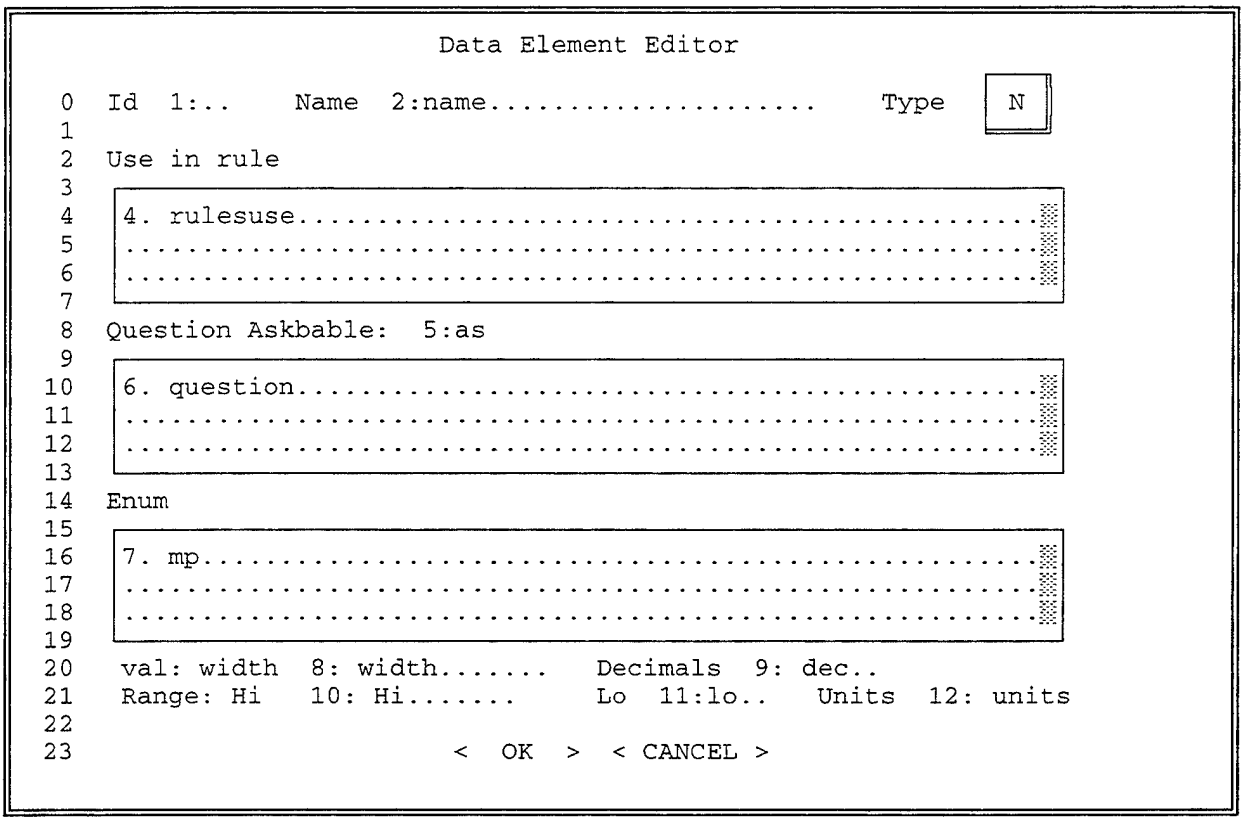
T. OBLIST.SCX **Last updated: 10/03/94 at 15:01**



Window name: W_obj
Coordinates: FROM 0,0 TO 0,61
Window options: FLOAT CLOSE MINIMIZE SHADOW

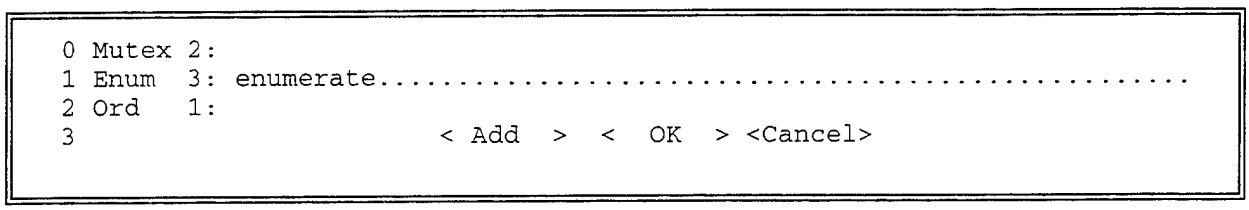
Name	Type	Picture
1: m.obj	List	"@&N"
2: m.buttons	Push button	"@*HT \<New;\<Ok;\<Cancel"

U. DDEDIT.SCX **Last updated: 10/03/94 at 15:01**



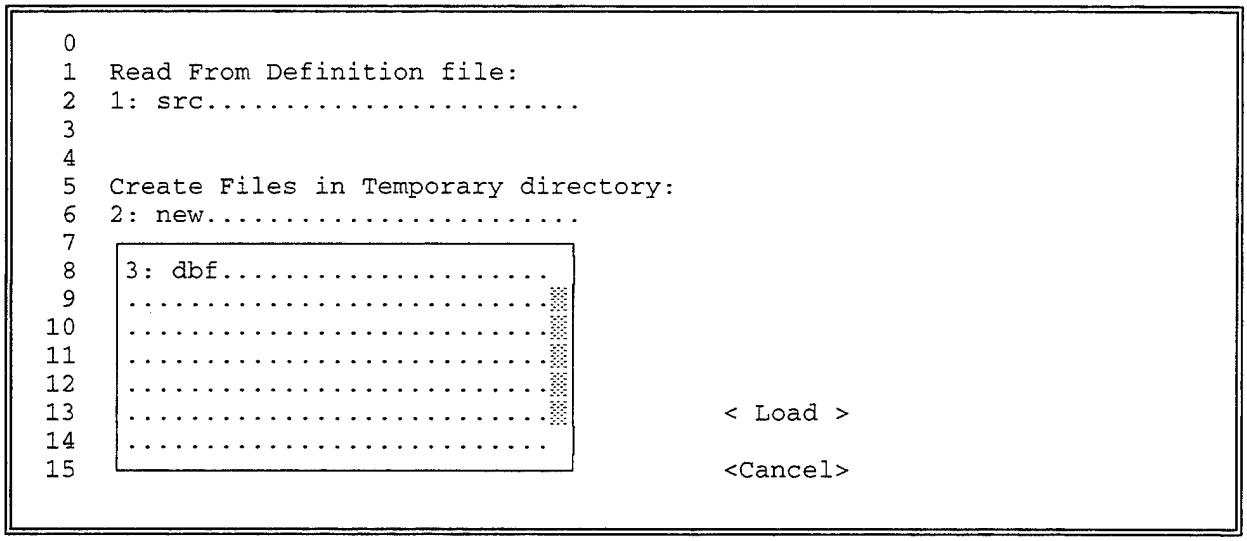
	Name	Type	Picture
1:	m.id	Field	
2:	m.name	Field	
3:	m.datatype	Popup	"@^ N;E;M;L"
4:	m.rulesuse	List	"@&N"
5:	m.askable	Field	
6:	m.question	Field	
7:	mp	List	"@&N"
8:	m.width	Field	
9:	m.dec	Field	
10:	m.hi	Field	
11:	m.lo	Field	
12:	m.units	Field	
13:	mbuttons	Push button	"@*HT OK;Cancel"

V. DDENUM.SCX **Last updated: 10/03/94 at 15:01**



	Name	Type	Picture
1:	m.ord	Field	
2:	m.mutex	Field	
3:	m.enumerate	Field	"@!"
4:	mbuttons	Push button	"@*HN Add;OK;Cancel"

W. KBLOAD.SCX **Last updated: 10/03/94 at 15:01**



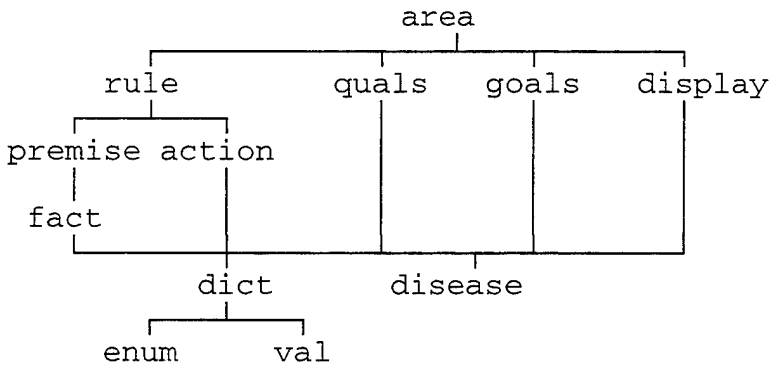
	Name	Type	Picture
1:	m.src	Field	
2:	m.new	Field	
3:	m.dbf	List	"@&N"
4:	m.load	Push button	"@*VN Load"
5:	mbuttonc	Push button	"@*VT Cancel"
6:	m.src	Field	
7:	m.new	Field	
8:	m.dbf	List	"@&N"
9:	m.load	Push button	"@*VT Load"
10:	mbuttonc	Push button	"@*VT Cancel"

Section IV. Data Dictionary. The system contains 14 databases.

```
DICT.DBF     -- Data element dictionary
ACTION.DBF   -- List of action for each rule
DISEASE.DBF  -- Disease element dictionary
ENUM.DBF     -- List of values for enumerate types
HELP.DBF     -- Fox command help text
KBHELP.DBF   -- Knowledge Base Command help text
PREMISE.DBF  -- List of premises for each rule
RULE.DBF     -- List of qualifiers for each knowledge base
VAL.DBF      -- Attributes for each numeric type
AREA.DBF     -- Diagnostic area or knowledge base
GOALS.DBF    -- List of goals for each knowledge base
DISPLAY.DBF  -- List of qualifiers shown for each knowledge base
QUALS.DBF    -- List of qualifiers for each knowledge base
FACT.DBF     -- Factors in rule premises
```

A. Knowledge Base Structure (relationship of database)

The following chart shows the structure of the knowledge base. This structure is used to set up relationships between the database and knowledge base:



Important facts about the knowledge base include:

The **AREA** file is the primary parent of all other files, as it relates to the inference engine. Therefore, selecting an area or knowledge base, means that the associated set of rules, qualifiers, goals and displays is also selected.

The **RULE** file is the parent of the PREMISE and ACTION files. Therefore, selecting a RULE, means that a set of PREMISE and ACTION clauses is also selected for that rule. The inference engine can select a particular rule and then retrieve corresponding PREMISE and ACTION clauses from the

child files. Since the RULE:PREMISE and RULE:ACTION relationships are one-to-many, several PREMISE and ACTION clauses can be associated with one rule.

The **PREMISE** file is the parent of the FACT file. Therefore, selecting a premise means that a set of fact clauses is also selected for that premise. When the inference engine evaluates a rule premise, it can retrieve the corresponding FACT clauses from the child file.

The **DICT** and **DISEASE** files are the children files of several of the knowledge base tables, including the QUALS, GOALS, DISPLAY, FACT, and ACTION files. The DICT and DISEASE files therefore play a central role in the knowledge base structure. These are the repositories for the basic data element definitions that are used in many places throughout the knowledge base.

The **DICT** file is the parent of the ENUM and VAL files. Each DICT element can have several ENUM entries associated with it,since this is a one-to-many relationship. Each DICT element can have one VAL entry associated with it, since this is a one-to-one relationship.

B. Database Structure Summary.

1. Structure for table/dbf: **DICT.DBF**
Number of data records : 256 Last updated : 06/17/93

Field	Field name	Type	Width	Dec	Start	End
1	ID	Numeric	4	0	1	4
2	NAME	Character	80	0	5	84
3	ALIAS	Character	20	0	85	104
4	DATATYPE	Character	1	0	105	105
5	ASKABLE	Logical	1	0	106	106
6	QUESTION	Character	100	0	107	206
7	PROMPT	Character	80	0	207	286
8	META	Logical	1	0	287	287
** Total **			288			

This table/dbf is associated with index file/tag(s):
: DICTID.IDX (ID)
: DICTNAME.IDX (NAME)
Used by: KB.SPR and DD.SPR

2. Structure for table/dbf: **DISEASE.DBF**
Number of data records : 51 Last updated : 06/10/93

Field	Field name	Type	Width	Dec	Start	End

1	ID	Numeric	5	0	1	5
2	NAME	Character	80	0	6	85
3	ALIAS	Character	20	0	86	105
4	DESCRIPT	Memo	10	0	106	115
5	TREATMENT	Memo	10	0	116	125
6	BRIEF	Memo	10	0	126	135
** Total **			136			

This table/dbf is associated with the memo file: DISEASE.FPT
This table/dbf is associated with index file/tag(s):
: DISID.IDX (ID)
: DISEASE.IDX (NAME)
Used by: KB.SPR

3. Structure for table/dbf: **ACTION.DBF**
Number of data records : 2344 Last updated : 01/12/93

Field	Field name	Type	Width	Dec	Start	End
1	CLAUSE	Numeric	5	0	1	5
2	OP	Character	2	0	6	7
3	OBJECT	Character	1	0	8	8
4	ID	Numeric	5	0	9	13
5	TAG	Character	1	0	14	14
6	VAL	Character	10	0	15	24
7	TEXT	Memo	10	0	25	34
** Total **			35			

This table/dbf is associated with the memo file: ACTION.FPT
This table/dbf appears to be associated with index file/tag(s):
: ACTION.IDX (CLAUSE)
Used by: KBLDR.PRG, KBDELETE.PRG, and KBEDIT.SPR

4. Structure for table/dbf: **ENUM.DBF**
Number of data records : 698 Last updated : 06/07/93

Field	Field name	Type	Width	Dec	Start	End
1	ID	Numeric	4	0	1	4
2	ORD	Numeric	2	0	5	6
3	MUTEX	Character	1	0	7	7
4	ENUMERATE	Character	80	0	8	87
5	REPORT	Character	80	0	88	167
** Total **			168			

This table/dbf appears to be associated with index file/tag(s):
: ENUM.IDX (ID)
Used by: KB.SPR and DD.SPR

5. Structure for table/dbf: **HELP.DBF**
Number of data records : 147 Last updated : 09/17/93

Field	Field name	Type	Width	Dec	Start	End

```
    1  TOPIC       Character      80       0        1        80
    2  DETAILS     Memo           10       0       81        90
    3  CLASS       Character      20       0       91       110
    4  ID          Numeric         5       0      111       115
    5  SOURCE      Character       1       0      116       116
** Total **                      117

This table/dbf is associated with the memo file: HELP.FPT
```

6. Structure for table/dbf: **KBHELP.DBF** Alias: HELP
Number of data records : 20 Last updated : 05/29/92

Field	Field name	Type	Width	Dec	Start	End
1	TOPIC	Character	30	0	1	30
2	DETAILS	Memo	10	0	31	40
3	CLASS	Character	20	0	41	60
4	ID	Numeric	5	0	61	65
5	SOURCE	Character	1	0	66	66
** Total **			67			

This table/dbf is associated with the memo file: KBHELP.FPT
Used by: KBLDR.PRG

7. Structure for table/dbf: **PREMISE.DBF**
Number of data records : 1809 Last updated : 01/12/93

Field	Field name	Type	Width	Dec	Start	End
1	CLAUSE	Numeric	5	0	1	5
2	OP	Character	1	0	6	6
3	FACT	Numeric	5	0	7	11
4	FACTR	Numeric	5	0	12	16
** Total **			17			

This table/dbf appears to be associated with index file/tag(s):
: PREMISE.IDX (CLAUSE)
Used by: KBLDR.PRG, KBDELETE.PRG, and KBEDIT.SPR

8. Structure for table/dbf: **RULE.DBF**
Number of data records : 564 Last updated : 06/01/93

Field	Field name	Type	Width	Dec	Start	End
1	RULE	Numeric	5	0	1	5
2	AREA	Numeric	4	0	6	9
3	SALIENCE	Numeric	3	0	10	12
4	PREMISE	Numeric	5	0	13	17
5	ACTION	Numeric	5	0	18	22
6	ELSE	Numeric	5	0	23	27
7	QUALS	Memo	10	0	28	37
8	GOALS	Memo	10	0	38	47
9	NOTE	Memo	10	0	48	57
10	EXPLAIN	Memo	10	0	58	67
** Total **			68			

This table/dbf is associated with the memo file: RULE.FPT
This table/dbf appears to be associated with index file/tag(s):
: RULEAREA.IDX (AREA)
: SALIENCE.IDX (SALIENCE)
: RULE.IDX (RULE)
Used by: KBLDR.PRG, KBDELETE.PRG, and KBEDIT.SPR

9. Structure for table/dbf: **VAL.DBF**
Number of data records : 19 Last updated : 04/29/93

Field	Field name	Type	Width	Dec	Start	End
1	ID	Numeric	4	0	1	4
2	WIDTH	Numeric	2	0	5	6
3	DEC	Numeric	1	0	7	7
4	LO	Numeric	9	2	8	16
5	HI	Numeric	9	2	17	25
6	UNITS	Character	10	0	26	35
** Total **			36			

This table/dbf appears to be associated with index file/tag(s):
: VAL.IDX (ID)
Used by: KB.SPR and DD.SPR

10. Structure for table/dbf: **AREA.DBF**
Number of data records : 4 Last updated : 06/17/93

Field	Field name	Type	Width	Dec	Start	End
1	AREA	Numeric	4	0	1	4
2	NAME	Character	30	0	5	34
3	INFERENCE	Numeric	1	0	35	35
4	METHOD	Numeric	1	0	36	36
5	ISDIAG	Numeric	1	0	37	37
6	SIGNON	Character	30	0	38	67
7	START	Character	30	0	68	97
8	FINISH	Character	30	0	98	127
9	THRESHOLD	Numeric	6	2	128	133
10	PROBABLE	Numeric	6	2	134	139
11	LIKELY	Numeric	6	2	140	145
12	RULES	Numeric	4	0	146	149
** Total **			150			

This table/dbf appears to be associated with index file/tag(s):
: AREA.IDX (AREA)
Used by: KB.SPR and DD.SPR

11. Structure for table/dbf: **GOALS.DBF**
Number of data records : 89 Last updated : 05/11/93

Field	Field name	Type	Width	Dec	Start	End
1	AREA	Numeric	4	0	1	4
2	OBJECT	Character	1	0	5	5
3	ID	Numeric	5	0	6	10
** Total **			11			

This table/dbf appears to be associated with index file/tag(s):
: GOALS.IDX (ID)
Used by: KB.SPR and DD.SPR

12. Structure for table/dbf: **DISPLAY.DBF**
Number of data records : 5 Last updated : 12/07/92

Field	Field name	Type	Width	Dec	Start	End
1	AREA	Numeric	5	0	1	5
2	ID	Numeric	5	0	6	10
3	OBJECT	Character	1	0	11	11
** Total **			12			

This table/dbf is not associated with index files/tags(s).
Used by: KB.SPR

13. Structure for table/dbf: **QUALS.DBF**
Number of data records : 292 Last updated : 01/12/93

Field	Field name	Type	Width	Dec	Start	End
1	AREA	Numeric	4	0	1	4
2	OBJECT	Character	1	0	5	5
3	ID	Numeric	5	0	6	10
4	RULES	Memo	10	0	11	20
5	RULESO	Memo	10	0	21	30
** Total **			31			

This table/dbf is associated with the memo file: QUALS.FPT
This table/dbf appears to be associated with index file/tag(s):
: QUALS.IDX (ID)
Used by: KB.SPR and DD.SPR

14. Structure for table/dbf: **FACT.DBF**
Number of data records : 1223 Last updated : 01/12/93

Field	Field name	Type	Width	Dec	Start	End
1	CLAUSE	Numeric	5	0	1	5
2	OP	Character	2	0	6	7
3	OBJECT	Character	1	0	8	8
4	ID	Numeric	5	0	9	13
5	TAG	Character	1	0	14	14
6	VAL	Character	10	0	15	24
7	TEXT	Memo	10	0	25	34
** Total **			35			

This table/dbf is associated with the memo file: FACT.FPT
This table/dbf appears to be associated with index file/tag(s):
: FACT.IDX (CLAUSE)
Used by: KBLDR.PRG, KBDELETE.PRG, and KBEDIT.SPR

C. Database Field Summary

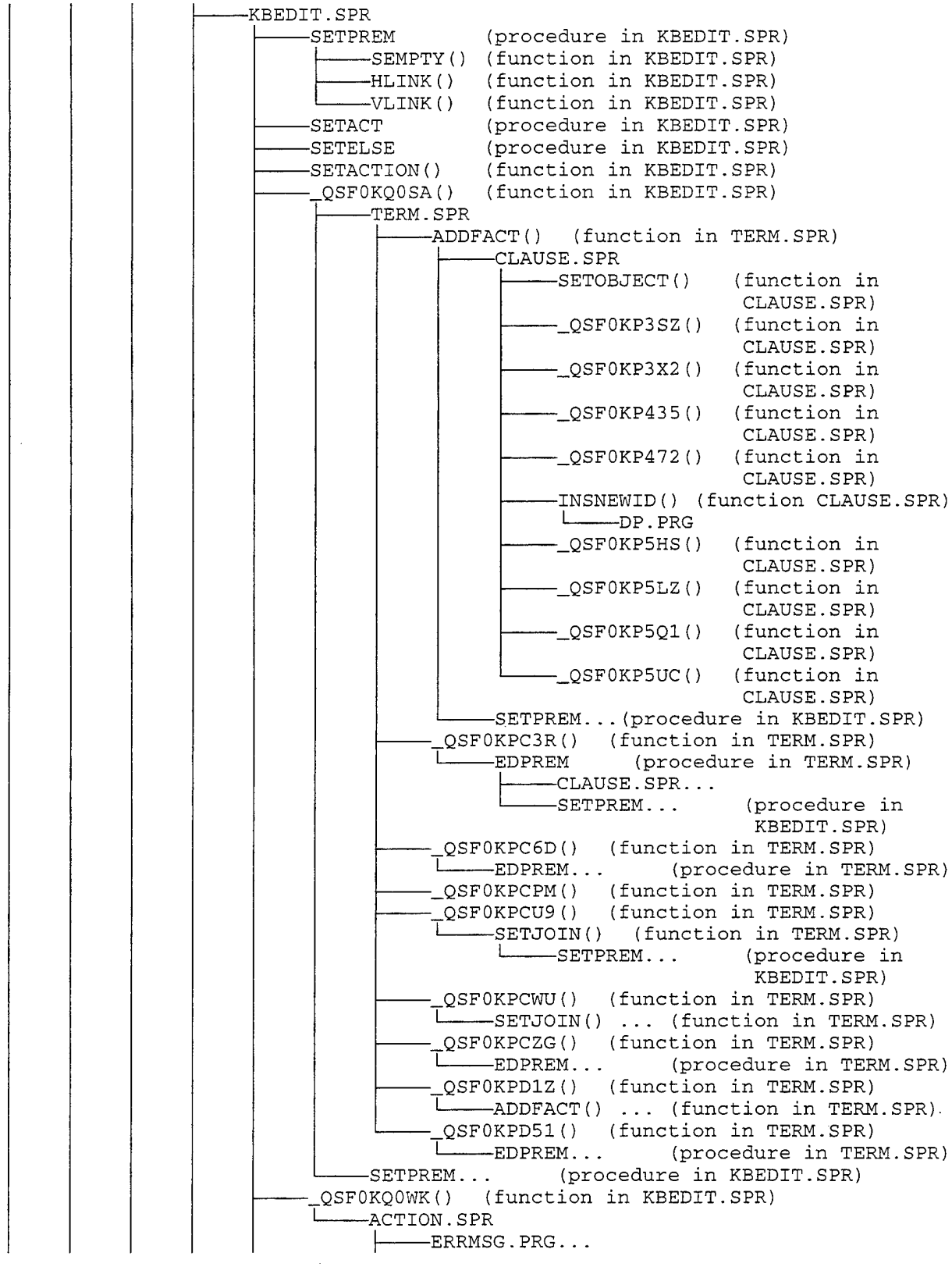
Field Name	Type	Len	Dec	Table/DBF
ACTION	N	5	0	RULE.DBF
ALIAS	C	20	0	DICT.DBF
ALIAS	C	20	0	DISEASE.DBF
AREA	N	4	0	AREA.DBF
AREA	N	5	0	DISPLAY.DBF
AREA	N	4	0	GOALS.DBF
AREA	N	4	0	QUALS.DBF
AREA	N	4	0	RULE.DBF
ASKABLE	L	1	0	DICT.DBF
BRIEF	M	10	0	DISEASE.DBF
CLASS	C	20	0	HELP.DBF
CLASS	C	20	0	KBHELP.DBF
CLAUSE	N	5	0	ACTION.DBF
CLAUSE	N	5	0	FACT.DBF
CLAUSE	N	5	0	PREMISE.DBF
DATATYPE	C	1	0	DICT.DBF
DEC	N	1	0	VAL.DBF
DESCRIPT	M	10	0	DISEASE.DBF
DETAILS	M	10	0	HELP.DBF
DETAILS	M	10	0	KBHELP.DBF
ELSE	N	5	0	RULE.DBF
ENUMERATE	C	80	0	ENUM.DBF
EXPLAIN	M	10	0	RULE.DBF
FACT	N	5	0	PREMISE.DBF
FACTR	N	5	0	PREMISE.DBF
FINISH	C	30	0	AREA.DBF
GOALS	M	10	0	RULE.DBF
HI	N	9	2	VAL.DBF
ID	N	5	0	ACTION.DBF
ID	N	4	0	DICT.DBF
ID	N	5	0	DISEASE.DBF
ID	N	5	0	DISPLAY.DBF
ID	N	4	0	ENUM.DBF
ID	N	5	0	FACT.DBF
ID	N	5	0	GOALS.DBF
ID	N	5	0	HELP.DBF
ID	N	5	0	KBHELP.DBF
ID	N	5	0	QUALS.DBF
ID	N	4	0	VAL.DBF
INFERENCE	N	1	0	AREA.DBF
ISDIAG	N	1	0	AREA.DBF
LIKELY	N	6	2	AREA.DBF
LO	N	9	2	VAL.DBF

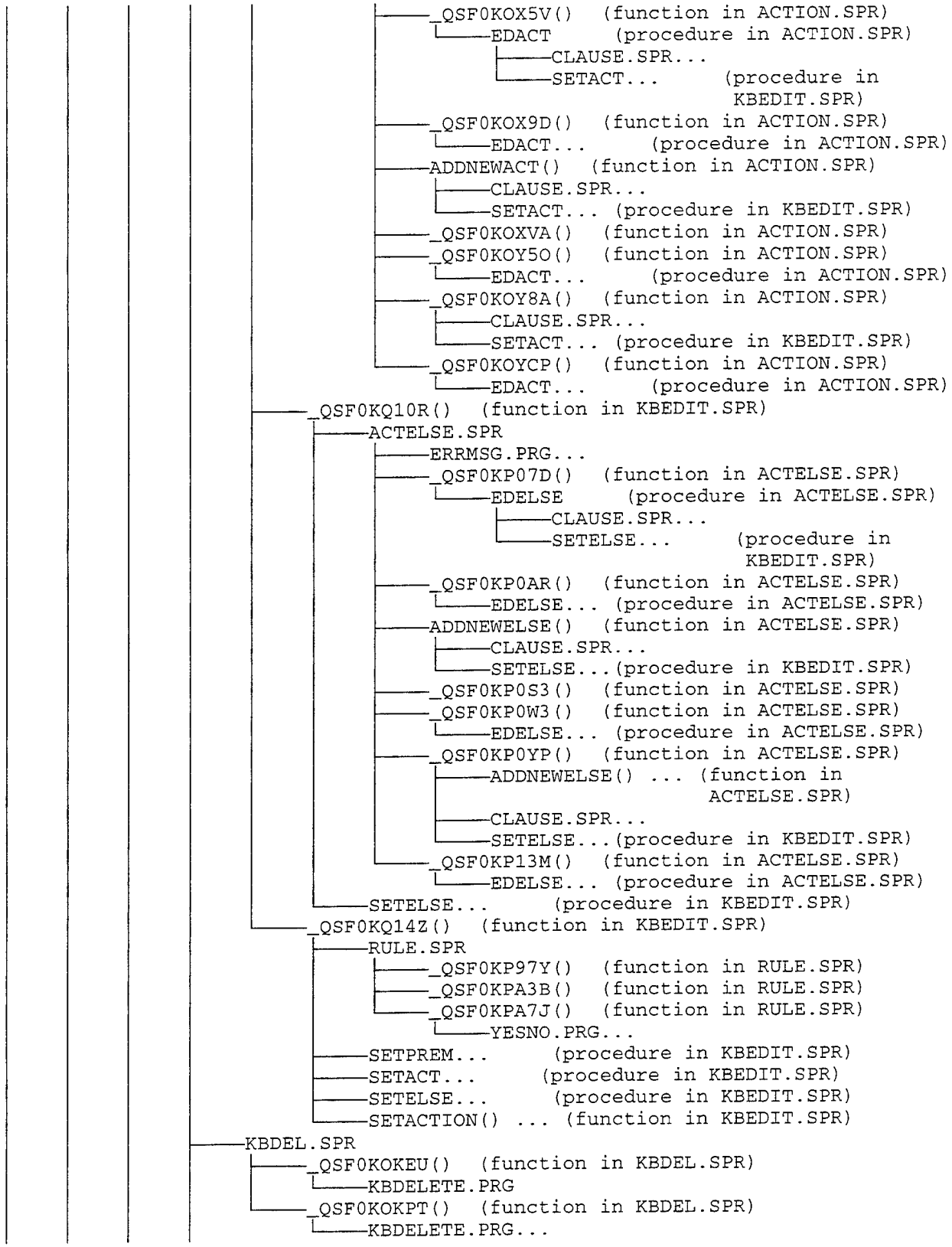
Field Name	Type	Len	Dec	Table/DBF
META	L	1	0	DICT.DBF
METHOD	N	1	0	AREA.DBF
MUTEX	C	1	0	ENUM.DBF
NAME	C	30	0	AREA.DBF
NAME	C	80	0	DICT.DBF
NAME	C	80	0	DISEASE.DBF
NOTE	M	10	0	RULE.DBF
OBJECT	C	1	0	ACTION.DBF
OBJECT	C	1	0	DISPLAY.DBF
OBJECT	C	1	0	FACT.DBF
OBJECT	C	1	0	GOALS.DBF
OBJECT	C	1	0	QUALS.DBF
OP	C	2	0	ACTION.DBF
OP	C	2	0	FACT.DBF
OP	C	1	0	PREMISE.DBF
ORD	N	2	0	ENUM.DBF
PREMISE	N	5	0	RULE.DBF
PROBABLE	N	6	2	AREA.DBF
PROMPT	C	80	0	DICT.DBF
QUALS	M	10	0	RULE.DBF
QUESTION	C	100	0	DICT.DBF
REPORT	C	80	0	ENUM.DBF
RULE	N	5	0	RULE.DBF
RULES	N	4	0	AREA.DBF
RULES	M	10	0	QUALS.DBF
RULESO	M	10	0	QUALS.DBF
SALIENCE	N	3	0	RULE.DBF
SIGNON	C	30	0	AREA.DBF
SOURCE	C	1	0	HELP.DBF
SOURCE	C	1	0	KBHELP.DBF
START	C	30	0	AREA.DBF
TAG	C	1	0	ACTION.DBF
TAG	C	1	0	FACT.DBF
TEXT	M	10	0	ACTION.DBF
TEXT	M	10	0	FACT.DBF
THRESHOLD	N	6	2	AREA.DBF
TOPIC	C	80	0	HELP.DBF
TOPIC	C	30	0	KBHELP.DBF
TREATMENT	M	10	0	DISEASE.DBF
UNITS	C	10	0	VAL.DBF
VAL	C	10	0	ACTION.DBF
VAL	C	10	0	FACT.DBF
WIDTH	N	2	0	VAL.DBF

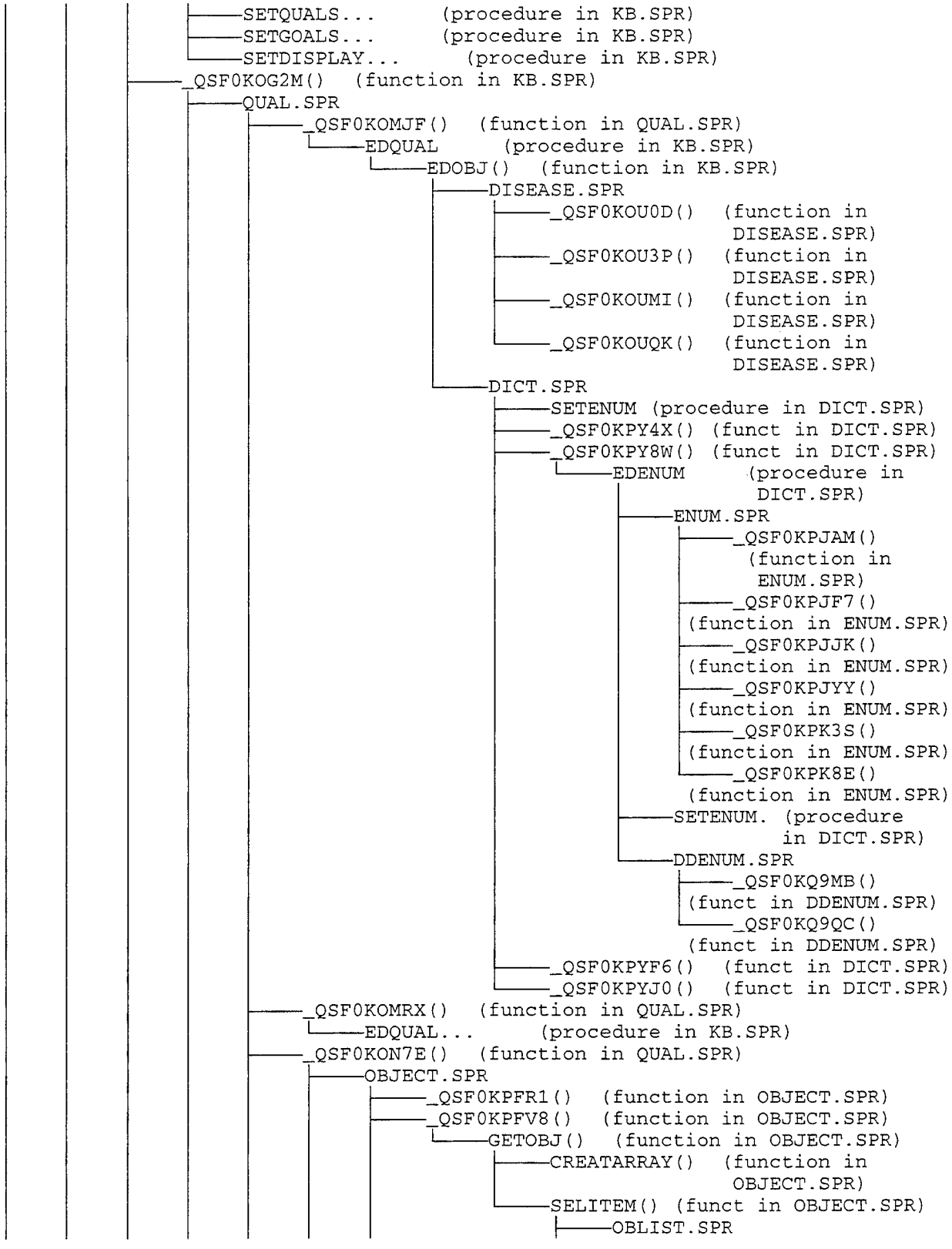
Section V. Tree Diagram. The tree diagram shows the correlate among the function and procedure. The diagram lists each program in The order in which it is used. Each program has a list of all functions called and where the procedure/function is stored.

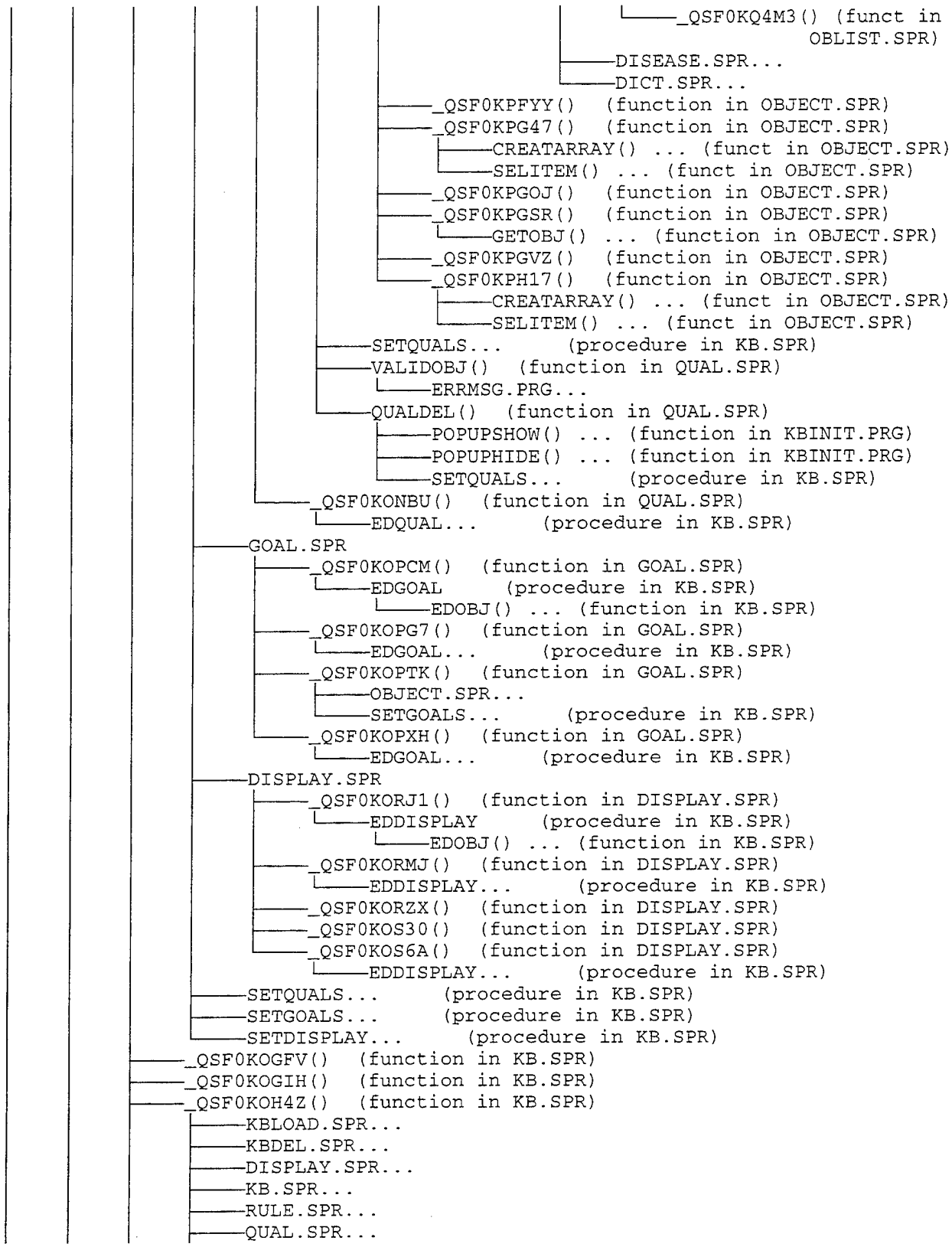
```
KBINIT.PRG
├──SETPATH()  (function in KBINIT.PRG)
├──KBMENU.MPR
│   ├──_QUIT()  (function in KBINIT.PRG)
│   │   └──ERRMSG.PRG
│   ├──BACKUP.PRG
│   │   └──BACKUP.SPR
│   │       ├──NEWDRIVEPOP   (procedure in BACKUP.SPR)
│   │       ├──NEWPATHPOP    (procedure in BACKUP.SPR)
│   │       ├──NEWFILEPOP    (procedure in BACKUP.SPR)
│   │       ├──_QSF0KO96C()  (function in BACKUP.SPR)
│   │       ├──_QSF0KO98Q()  (function in BACKUP.SPR)
│   │       │   ├──YESNO.PRG
│   │       │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │       │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KO9EC()  (function in BACKUP.SPR)
│   │       │   ├──PATHSTRING        (procedure in BACKUP.SPR)
│   │       │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │       │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KO9HK()  (function in BACKUP.SPR)
│   │       │   ├──YESNO.PRG...
│   │       │   └──PKZIP     (procedure in BACKUP.SPR)
│   │       │       └──PATHSTRING...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KO9LV()  (function in BACKUP.SPR)
│   │       │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │       │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KOAG9()  (function in BACKUP.SPR)
│   │       ├──_QSF0KOAID()  (function in BACKUP.SPR)
│   │       │   ├──YESNO.PRG...
│   │       │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │       │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KOANU()  (function in BACKUP.SPR)
│   │       │   ├──PATHSTRING...     (procedure in BACKUP.SPR)
│   │       │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │       │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   │       ├──_QSF0KOAQV()  (function in BACKUP.SPR)
│   │       │   ├──YESNO.PRG...
│   │       │   └──PKZIP...      (procedure in BACKUP.SPR)
│   │       └──_QSF0KOAUY()  (function in BACKUP.SPR)
│   │           ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│   │           └──NEWFILEPOP...     (procedure in BACKUP.SPR)
│   └──RESTORE.PRG
│       └──RESTORE.SPR
│           ├──NEWDRIVEPOP...     (procedure in BACKUP.SPR)
│           ├──NEWPATHPOP...      (procedure in BACKUP.SPR)
│           ├──NEWFILEPOP...      (procedure in BACKUP.SPR)
│           ├──_QSF0KPMPZ()  (function in RESTORE.SPR)
│           ├──_QSF0KPMS0()  (function in RESTORE.SPR)
│           │   ├──YESNO.PRG...
│           │   ├──NEWPATHPOP...     (procedure in BACKUP.SPR)
│           │   └──NEWFILEPOP...     (procedure in BACKUP.SPR)
```

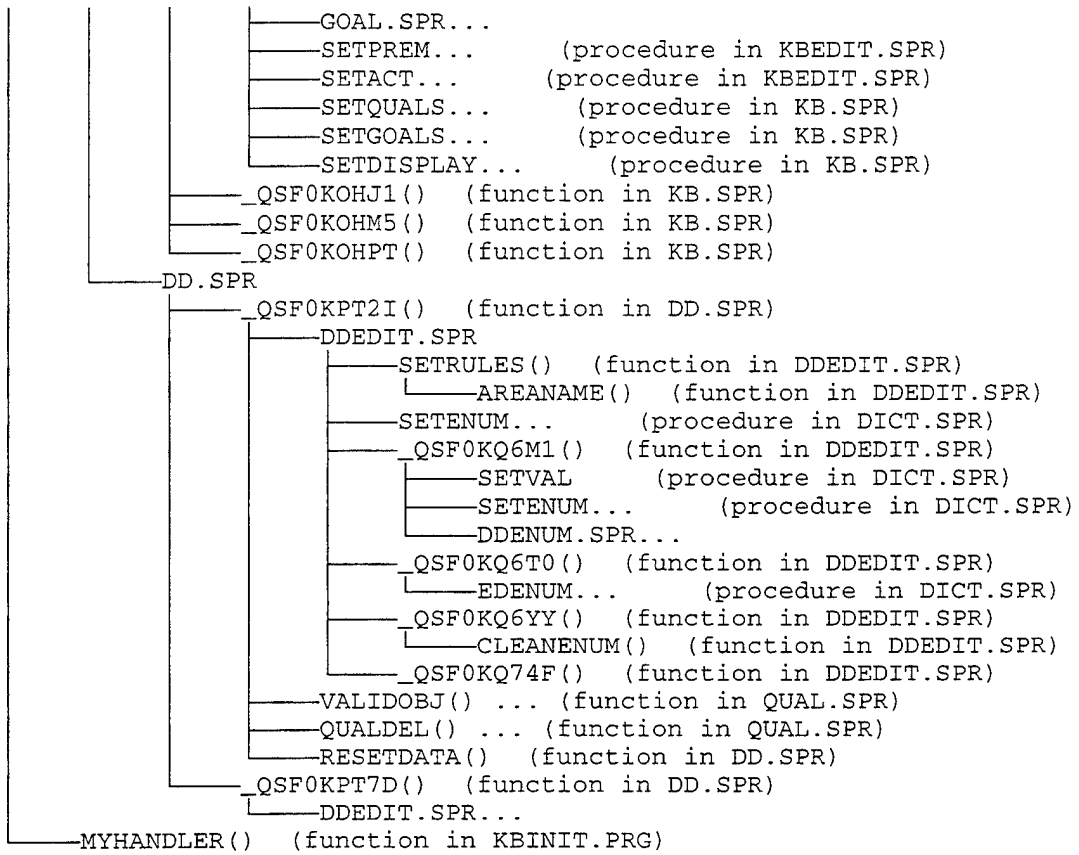
```
        ├─────_QSF0KPMXI()  (function in RESTORE.SPR)
        │       ├─────PATHSTRING...      (procedure in BACKUP.SPR)
        │       ├─────NEWPATHPOP...      (procedure in BACKUP.SPR)
        │       └─────NEWFILEPOP...      (procedure in BACKUP.SPR)
        ├─────_QSF0KPN0J()  (function in RESTORE.SPR)
        │       └─────PKUNZIP     (procedure in RESTORE.SPR)
        │             └─────PATHSTRING...      (procedure in BACKUP.SPR)
        ├─────_QSF0KPN3T()  (function in RESTORE.SPR)
        │       ├─────NEWPATHPOP...      (procedure in BACKUP.SPR)
        │       └─────NEWFILEPOP...      (procedure in BACKUP.SPR)
        ├─────_QSF0KPN7E()  (function in RESTORE.SPR)
        ├─────_QSF0KPNY5()  (function in RESTORE.SPR)
        ├─────_QSF0KPO0A()  (function in RESTORE.SPR)
        │       ├─────YESNO.PRG...
        │       ├─────NEWPATHPOP...      (procedure in BACKUP.SPR)
        │       └─────NEWFILEPOP...      (procedure in BACKUP.SPR)
        ├─────_QSF0KPO5T()  (function in RESTORE.SPR)
        │       ├─────PATHSTRING...      (procedure in BACKUP.SPR)
        │       ├─────NEWPATHPOP...      (procedure in BACKUP.SPR)
        │       └─────NEWFILEPOP...      (procedure in BACKUP.SPR)
        ├─────_QSF0KPO8T()  (function in RESTORE.SPR)
        │       └─────PKUNZIP...     (procedure in RESTORE.SPR)
        ├─────_QSF0KPOBY()  (function in RESTORE.SPR)
        │       ├─────NEWPATHPOP...      (procedure in BACKUP.SPR)
        │       └─────NEWFILEPOP...      (procedure in BACKUP.SPR)
        └─────_QSF0KPOFG()  (function in RESTORE.SPR)
├─────KB.SPR
    ├─────SETQUALS      (procedure in KB.SPR)
    ├─────SETGOALS      (procedure in KB.SPR)
    ├─────SETDISPLAY    (procedure in KB.SPR)
    ├─────_QSF0KOFX9()  (function in KB.SPR)
    │       ├─────KBLOAD.SPR
    │       │       ├─────_QSF0KQBNK()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK     (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQBT0()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQC1J()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQC42()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQC6T()  (function in KBLOAD.SPR)
    │       │       │       ├─────CHECKFILE()  (function in KBLOAD.SPR)
    │       │       │       │       └─────YESNO.PRG...
    │       │       │       ├─────POPUPSHOW()  (function in KBINIT.PRG)
    │       │       │       ├─────KBLDR.PRG
    │       │       │       └─────POPUPHIDE()  (function in KBINIT.PRG)
    │       │       ├─────_QSF0KQCP2()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQCRO()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQCUA()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       ├─────_QSF0KQCWT()  (function in KBLOAD.SPR)
    │       │       │       └─────LOADOK...      (procedure in KBLOAD.SPR)
    │       │       └─────_QSF0KQCZG()  (function in KBLOAD.SPR)
    │       │               └─────KBLDR.PRG...
```


KBEDIT.SPR
SETPREM (procedure in KBEDIT.SPR)
SEMPTY() (function in KBEDIT.SPR)
HLINK() (function in KBEDIT.SPR)
VLINK() (function in KBEDIT.SPR)
SETACT (procedure in KBEDIT.SPR)
SETELSE (procedure in KBEDIT.SPR)
SETACTION() (function in KBEDIT.SPR)
_QSF0KQ0SA() (function in KBEDIT.SPR)
TERM.SPR
ADDFACT() (function in TERM.SPR)
CLAUSE.SPR
SETOBJECT() (function in CLAUSE.SPR)
_QSF0KP3SZ() (function in CLAUSE.SPR)
_QSF0KP3X2() (function in CLAUSE.SPR)
_QSF0KP435() (function in CLAUSE.SPR)
_QSF0KP472() (function in CLAUSE.SPR)
INSNEWID() (function CLAUSE.SPR)
DP.PRG
_QSF0KP5HS() (function in CLAUSE.SPR)
_QSF0KP5LZ() (function in CLAUSE.SPR)
_QSF0KP5Q1() (function in CLAUSE.SPR)
_QSF0KP5UC() (function in CLAUSE.SPR)
SETPREM...(procedure in KBEDIT.SPR)
_QSF0KPC3R() (function in TERM.SPR)
EDPREM (procedure in TERM.SPR)
CLAUSE.SPR...
SETPREM... (procedure in KBEDIT.SPR)
_QSF0KPC6D() (function in TERM.SPR)
EDPREM... (procedure in TERM.SPR)
_QSF0KPCPM() (function in TERM.SPR)
_QSF0KPCU9() (function in TERM.SPR)
SETJOIN() (function in TERM.SPR)
SETPREM... (procedure in KBEDIT.SPR)
_QSF0KPCWU() (function in TERM.SPR)
SETJOIN() ... (function in TERM.SPR)
_QSF0KPCZG() (function in TERM.SPR)
EDPREM... (procedure in TERM.SPR)
_QSF0KPD1Z() (function in TERM.SPR)
ADDFACT() ... (function in TERM.SPR)
_QSF0KPD51() (function in TERM.SPR)
EDPREM... (procedure in TERM.SPR)
SETPREM... (procedure in KBEDIT.SPR)
_QSF0KQ0WK() (function in KBEDIT.SPR)
ACTION.SPR
ERRMSG.PRG...


_QSF0KOX5V() (function in ACTION.SPR)
EDACT (procedure in ACTION.SPR)
CLAUSE.SPR...
SETACT... (procedure in KBEDIT.SPR)
_QSF0KOX9D() (function in ACTION.SPR)
EDACT... (procedure in ACTION.SPR)
ADDNEWACT() (function in ACTION.SPR)
CLAUSE.SPR...
SETACT... (procedure in KBEDIT.SPR)
_QSF0KOXVA() (function in ACTION.SPR)
_QSF0KOY5O() (function in ACTION.SPR)
EDACT... (procedure in ACTION.SPR)
_QSF0KOY8A() (function in ACTION.SPR)
CLAUSE.SPR...
SETACT... (procedure in KBEDIT.SPR)
_QSF0KOYCP() (function in ACTION.SPR)
EDACT... (procedure in ACTION.SPR)
_QSF0KQ10R() (function in KBEDIT.SPR)
ACTELSE.SPR
ERRMSG.PRG...
_QSF0KP07D() (function in ACTELSE.SPR)
EDELSE (procedure in ACTELSE.SPR)
CLAUSE.SPR...
SETELSE... (procedure in KBEDIT.SPR)
_QSF0KP0AR() (function in ACTELSE.SPR)
EDELSE... (procedure in ACTELSE.SPR)
ADDNEWELSE() (function in ACTELSE.SPR)
CLAUSE.SPR...
SETELSE...(procedure in KBEDIT.SPR)
_QSF0KP0S3() (function in ACTELSE.SPR)
_QSF0KP0W3() (function in ACTELSE.SPR)
EDELSE... (procedure in ACTELSE.SPR)
_QSF0KP0YP() (function in ACTELSE.SPR)
ADDNEWELSE() ... (function in ACTELSE.SPR)
CLAUSE.SPR...
SETELSE...(procedure in KBEDIT.SPR)
_QSF0KP13M() (function in ACTELSE.SPR)
EDELSE... (procedure in ACTELSE.SPR)
SETELSE... (procedure in KBEDIT.SPR)
_QSF0KQ14Z() (function in KBEDIT.SPR)
RULE.SPR
_QSF0KP97Y() (function in RULE.SPR)
_QSF0KPA3B() (function in RULE.SPR)
_QSF0KPA7J() (function in RULE.SPR)
YESNO.PRG...
SETPREM... (procedure in KBEDIT.SPR)
SETACT... (procedure in KBEDIT.SPR)
SETELSE... (procedure in KBEDIT.SPR)
SETACTION() ... (function in KBEDIT.SPR)
KBDEL.SPR
_QSF0KOKEU() (function in KBDEL.SPR)
KBDELETE.PRG
_QSF0KOKPT() (function in KBDEL.SPR)
KBDELETE.PRG...


SETQUALS... (procedure in KB.SPR)
SETGOALS... (procedure in KB.SPR)
SETDISPLAY... (procedure in KB.SPR)
_QSF0KOG2M() (function in KB.SPR)
QUAL.SPR
_QSF0KOMJF() (function in QUAL.SPR)
EDQUAL (procedure in KB.SPR)
EDOBJ() (function in KB.SPR)
DISEASE.SPR
_QSF0KOU0D() (function in DISEASE.SPR)
_QSF0KOU3P() (function in DISEASE.SPR)
_QSF0KOUMI() (function in DISEASE.SPR)
_QSF0KOUQK() (function in DISEASE.SPR)
DICT.SPR
SETENUM (procedure in DICT.SPR)
_QSF0KPY4X() (funct in DICT.SPR)
_QSF0KPY8W() (funct in DICT.SPR)
EDENUM (procedure in DICT.SPR)
ENUM.SPR
_QSF0KPJAM() (function in ENUM.SPR)
_QSF0KPJF7() (function in ENUM.SPR)
_QSF0KPJJK() (function in ENUM.SPR)
_QSF0KPJYY() (function in ENUM.SPR)
_QSF0KPK3S() (function in ENUM.SPR)
_QSF0KPK8E() (function in ENUM.SPR)
SETENUM. (procedure in DICT.SPR)
DDENUM.SPR
_QSF0KQ9MB() (funct in DDENUM.SPR)
_QSF0KQ9QC() (funct in DDENUM.SPR)
_QSF0KPYF6() (funct in DICT.SPR)
_QSF0KPYJ0() (funct in DICT.SPR)
_QSF0KOMRX() (function in QUAL.SPR)
EDQUAL... (procedure in KB.SPR)
_QSF0KON7E() (function in QUAL.SPR)
OBJECT.SPR
_QSF0KPFR1() (function in OBJECT.SPR)
_QSF0KPFV8() (function in OBJECT.SPR)
GETOBJ() (function in OBJECT.SPR)
CREATARRAY() (function in OBJECT.SPR)
SELITEM() (funct in OBJECT.SPR)
OBLIST.SPR


_QSF0KQ4M3() (funct in OBLIST.SPR)
DISEASE.SPR...
DICT.SPR...
_QSF0KPFYY() (function in OBJECT.SPR)
_QSF0KPG47() (function in OBJECT.SPR)
CREATARRAY() ... (funct in OBJECT.SPR)
SELITEM() ... (funct in OBJECT.SPR)
_QSF0KPGOJ() (function in OBJECT.SPR)
_QSF0KPGSR() (function in OBJECT.SPR)
GETOBJ() ... (function in OBJECT.SPR)
_QSF0KPGVZ() (function in OBJECT.SPR)
_QSF0KPH17() (function in OBJECT.SPR)
CREATARRAY() ... (funct in OBJECT.SPR)
SELITEM() ... (funct in OBJECT.SPR)
SETQUALS... (procedure in KB.SPR)
VALIDOBJ() (function in QUAL.SPR)
ERRMSG.PRG...
QUALDEL() (function in QUAL.SPR)
POPUPSHOW() ... (function in KBINIT.PRG)
POPUPHIDE() ... (function in KBINIT.PRG)
SETQUALS... (procedure in KB.SPR)
_QSF0KONBU() (function in QUAL.SPR)
EDQUAL... (procedure in KB.SPR)
GOAL.SPR
_QSF0KOPCM() (function in GOAL.SPR)
EDGOAL (procedure in KB.SPR)
EDOBJ() ... (function in KB.SPR)
_QSF0KOPG7() (function in GOAL.SPR)
EDGOAL... (procedure in KB.SPR)
_QSF0KOPTK() (function in GOAL.SPR)
OBJECT.SPR...
SETGOALS... (procedure in KB.SPR)
_QSF0KOPXH() (function in GOAL.SPR)
EDGOAL... (procedure in KB.SPR)
DISPLAY.SPR
_QSF0KORJ1() (function in DISPLAY.SPR)
EDDISPLAY (procedure in KB.SPR)
EDOBJ() ... (function in KB.SPR)
_QSF0KORMJ() (function in DISPLAY.SPR)
EDDISPLAY... (procedure in KB.SPR)
_QSF0KORZX() (function in DISPLAY.SPR)
_QSF0KOS30() (function in DISPLAY.SPR)
_QSF0KOS6A() (function in DISPLAY.SPR)
EDDISPLAY... (procedure in KB.SPR)
SETQUALS... (procedure in KB.SPR)
SETGOALS... (procedure in KB.SPR)
SETDISPLAY... (procedure in KB.SPR)
_QSF0KOGFV() (function in KB.SPR)
_QSF0KOGIH() (function in KB.SPR)
_QSF0KOH4Z() (function in KB.SPR)
KBLOAD.SPR...
KBDEL.SPR...
DISPLAY.SPR...
KB.SPR...
RULE.SPR...
QUAL.SPR...


GOAL.SPR...
SETPREM... (procedure in KBEDIT.SPR)
SETACT... (procedure in KBEDIT.SPR)
SETQUALS... (procedure in KB.SPR)
SETGOALS... (procedure in KB.SPR)
SETDISPLAY... (procedure in KB.SPR)
_QSF0KOHJ1() (function in KB.SPR)
_QSF0KOHM5() (function in KB.SPR)
_QSF0KOHPT() (function in KB.SPR)
DD.SPR
_QSF0KPT2I() (function in DD.SPR)
DDEDIT.SPR
SETRULES() (function in DDEDIT.SPR)
AREANAME() (function in DDEDIT.SPR)
SETENUM... (procedure in DICT.SPR)
_QSF0KQ6M1() (function in DDEDIT.SPR)
SETVAL (procedure in DICT.SPR)
SETENUM... (procedure in DICT.SPR)
DDENUM.SPR...
_QSF0KQ6T0() (function in DDEDIT.SPR)
EDENUM... (procedure in DICT.SPR)
_QSF0KQ6YY() (function in DDEDIT.SPR)
CLEANENUM() (function in DDEDIT.SPR)
_QSF0KQ74F() (function in DDEDIT.SPR)
VALIDOBJ() ... (function in QUAL.SPR)
QUALDEL() ... (function in QUAL.SPR)
RESETDATA() (function in DD.SPR)
_QSF0KPT7D() (function in DD.SPR)
DDEDIT.SPR...
MYHANDLER() (function in KBINIT.PRG)

Section VI. Procedure and Function Summary. The system contains 24 procedure files: KBINIT.PRG, KBLOAD.SPR, KB.SPR, KBDEL.SPR, QUAL.SPR, GOAL.SPR, DISPLAY.SPR, DISEASE.SPR, ACTION.SPR, ACTELSE.SPR, CLAUSE.SPR, RULE.SPR, TERM.SPR, OBJECT.SPR, ENUM.SPR, RESTORE.SPR, DD.SPR, PLAN.SPR, DICT.SPR, KBEDIT.SPR, OBLIST.SPR, DDEDIT.SPR, DDENUM.SPR, and BACKUP.SPR

1. KBINIT.PRG

```
Contains: MYHANDLER()                    (Params: none)
   Called by: KBINIT.PRG
Contains: _QUIT()                        (Params: none)
   Called by: KBMENU.MPR
       Calls: ERRMSG.PRG
Contains: POPUPSHOW()                    (Params: ERRSTR)
   Called by: QUALDEL()                  (function in QUAL.SPR)
   Called by: _QSF0KQC6T()               (function in KBLOAD.SPR)
Contains: POPUPHIDE()                    (Params: W)
   Called by: QUALDEL()                  (function in QUAL.SPR)
   Called by: _QSF0KQC6T()               (function in KBLOAD.SPR)
Contains: SETPATH()                      (Params: NEWPATH)
   Called by: KBINIT.PRG
```

2. KBLOAD.SPR

```
Contains: _QSF0KQBNK()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQBT0()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQC1J()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQC42()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQC6T()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: CHECKFILE()                (function in KBLOAD.SPR)
       Calls: POPUPSHOW()                (function in KBINIT.PRG)
       Calls: KBLDR.PRG
       Calls: POPUPHIDE()                (function in KBINIT.PRG)
Contains: _QSF0KQCP2()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQCRO()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQCUA()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQCWT()                   (Params: none)
   Called by: KBLOAD.SPR
       Calls: LOADOK                     (procedure in KBLOAD.SPR)
Contains: _QSF0KQCZG()                   (Params: none)
```

```
   Called by: KBLOAD.SPR
       Calls: KBLDR.PRG
Contains: LOADOK                          (Params: none)
   Called by: KBLOAD.SPR
   Called by: _QSF0KQBNK()                (function in KBLOAD.SPR)
   Called by: _QSF0KQBT0()                (function in KBLOAD.SPR)
   Called by: _QSF0KQC1J()                (function in KBLOAD.SPR)
   Called by: _QSF0KQC42()                (function in KBLOAD.SPR)
   Called by: _QSF0KQCP2()                (function in KBLOAD.SPR)
   Called by: _QSF0KQCRO()                (function in KBLOAD.SPR)
   Called by: _QSF0KQCUA()                (function in KBLOAD.SPR)
   Called by: _QSF0KQCWT()                (function in KBLOAD.SPR)
Contains: CHECKFILE()                     (Params: M.NEW)
   Called by: _QSF0KQC6T()                (function in KBLOAD.SPR)
       Calls: YESNO.PRG
```

3. KB.SPR

```
Contains: _QSF0KOFX9()                    (Params: none)
   Called by: KB.SPR
       Calls: KBLOAD.SPR
       Calls: KBEDIT.SPR
       Calls: KBDEL.SPR
       Calls: SETQUALS                    (procedure in KB.SPR)
       Calls: SETGOALS                    (procedure in KB.SPR)
       Calls: SETDISPLAY                  (procedure in KB.SPR)
Contains: _QSF0KOG2M()                    (Params: none)
   Called by: KB.SPR
       Calls: QUAL.SPR
       Calls: GOAL.SPR
       Calls: DISPLAY.SPR
       Calls: SETQUALS                    (procedure in KB.SPR)
       Calls: SETGOALS                    (procedure in KB.SPR)
       Calls: SETDISPLAY                  (procedure in KB.SPR)
Contains: _QSF0KOGFV()                    (Params: none)
   Called by: KB.SPR
Contains: _QSF0KOGIH()                    (Params: none)
   Called by: KB.SPR
Contains: _QSF0KOH4Z()                    (Params: none)
   Called by: KB.SPR
       Calls: KBLOAD.SPR
       Calls: KBDEL.SPR
       Calls: DISPLAY.SPR
       Calls: KB.SPR
       Calls: RULE.SPR
       Calls: QUAL.SPR
       Calls: GOAL.SPR
       Calls: SETPREM                     (procedure in KBEDIT.SPR)
       Calls: SETACT                      (procedure in KBEDIT.SPR)
       Calls: SETQUALS                    (procedure in KB.SPR)
       Calls: SETGOALS                    (procedure in KB.SPR)
       Calls: SETDISPLAY                  (procedure in KB.SPR)
Contains: _QSF0KOHJ1()                    (Params: none)
   Called by: KB.SPR
Contains: _QSF0KOHM5()                    (Params: none)
   Called by: KB.SPR
Contains: _QSF0KOHPT()                    (Params: none)
```

```
   Called by: KB.SPR
Contains: SETQUALS                    (Params: none)
   Called by: KB.SPR
   Called by: _QSF0KOFX9()            (function in KB.SPR)
   Called by: _QSF0KOG2M()            (function in KB.SPR)
   Called by: _QSF0KOH4Z()            (function in KB.SPR)
   Called by: _QSF0KON7E()            (function in QUAL.SPR)
   Called by: QUALDEL()               (function in QUAL.SPR)
Contains: SETGOALS                    (Params: none)
   Called by: KB.SPR
   Called by: _QSF0KOFX9()            (function in KB.SPR)
   Called by: _QSF0KOG2M()            (function in KB.SPR)
   Called by: _QSF0KOH4Z()            (function in KB.SPR)
   Called by: _QSF0KOPTK()            (function in GOAL.SPR)
   Called by: _QSF0KPV8R()            (function in PLAN.SPR)
   Called by: _QSF0KPVQB()            (function in PLAN.SPR)
Contains: SETDISPLAY                  (Params: none)
   Called by: KB.SPR
   Called by: _QSF0KOFX9()            (function in KB.SPR)
   Called by: _QSF0KOG2M()            (function in KB.SPR)
   Called by: _QSF0KOH4Z()            (function in KB.SPR)
Contains: EDGOAL                      (Params: none)
   Called by: _QSF0KOPCM()            (function in GOAL.SPR)
   Called by: _QSF0KOPG7()            (function in GOAL.SPR)
   Called by: _QSF0KOPXH()            (function in GOAL.SPR)
   Called by: _QSF0KPVCR()            (function in PLAN.SPR)
   Called by: _QSF0KPVW8()            (function in PLAN.SPR)
       Calls: EDOBJ()                 (function in KB.SPR)
Contains: EDQUAL                      (Params: none)
   Called by: _QSF0KOMJF()            (function in QUAL.SPR)
   Called by: _QSF0KOMRX()            (function in QUAL.SPR)
   Called by: _QSF0KONBU()            (function in QUAL.SPR)
       Calls: EDOBJ()                 (function in KB.SPR)
Contains: EDDISPLAY                   (Params: none)
   Called by: _QSF0KORJ1()            (function in DISPLAY.SPR)
   Called by: _QSF0KORMJ()            (function in DISPLAY.SPR)
   Called by: _QSF0KOS6A()            (function in DISPLAY.SPR)
       Calls: EDOBJ()                 (function in KB.SPR)
Contains: EDOBJ()                     (Params: MOBJ, MID)
   Called by: EDGOAL                  (procedure in KB.SPR)
   Called by: EDQUAL                  (procedure in KB.SPR)
   Called by: EDDISPLAY               (procedure in KB.SPR)
       Calls: DISEASE.SPR
       Calls: DICT.SPR
```

4. KBDEL.SPR

```
Contains: _QSF0KOKEU()                (Params: none)
   Called by: KBDEL.SPR
       Calls: KBDELETE.PRG
Contains: _QSF0KOKPT()                (Params: none)
   Called by: KBDEL.SPR
       Calls: KBDELETE.PRG
```

5. QUAL.SPR

```
Contains: _QSF0KOMJF()                (Params: none)
```

```
   Called by: QUAL.SPR
       Calls: EDQUAL                    (procedure in KB.SPR)
Contains: _QSF0KOMRX()                  (Params: none)
   Called by: QUAL.SPR
       Calls: EDQUAL                    (procedure in KB.SPR)
Contains: _QSF0KON7E()                  (Params: none)
   Called by: QUAL.SPR
       Calls: OBJECT.SPR
       Calls: SETQUALS                  (procedure in KB.SPR)
       Calls: VALIDOBJ()                (function in QUAL.SPR)
       Calls: QUALDEL()                 (function in QUAL.SPR)
Contains: _QSF0KONBU()                  (Params: none)
   Called by: QUAL.SPR
       Calls: EDQUAL                    (procedure in KB.SPR)
Contains: POPUPSHOW()                   (Params: ERRSTR)
   Called by: QUALDEL()                 (function in QUAL.SPR)
   Called by: _QSF0KQC6T()              (function in KBLOAD.SPR)
Contains: POPUPHIDE()                   (Params: W)
   Called by: QUALDEL()                 (function in QUAL.SPR)
   Called by: _QSF0KQC6T()              (function in KBLOAD.SPR)
Contains: VALIDOBJ()                    (Params: none)
   Called by: _QSF0KON7E()              (function in QUAL.SPR)
   Called by: _QSF0KPT2I()              (function in DD.SPR)
       Calls: ERRMSG.PRG
Contains: QUALDEL()                     (Params: none)
   Called by: _QSF0KON7E()              (function in QUAL.SPR)
   Called by: _QSF0KPT2I()              (function in DD.SPR)
       Calls: POPUPSHOW()               (function in KBINIT.PRG)
       Calls: POPUPHIDE()               (function in KBINIT.PRG)
       Calls: SETQUALS                  (procedure in KB.SPR)
```

6. GOAL.SPR

```
Contains: _QSF0KOPCM()                  (Params: none)
   Called by: GOAL.SPR
       Calls: EDGOAL                    (procedure in KB.SPR)
Contains: _QSF0KOPG7()                  (Params: none)
   Called by: GOAL.SPR
       Calls: EDGOAL                    (procedure in KB.SPR)
Contains: _QSF0KOPTK()                  (Params: none)
   Called by: GOAL.SPR
       Calls: OBJECT.SPR
       Calls: SETGOALS                  (procedure in KB.SPR)
Contains: _QSF0KOPXH()                  (Params: none)
   Called by: GOAL.SPR
       Calls: EDGOAL                    (procedure in KB.SPR)
```

7. DISPLAY.SPR

```
Contains: _QSF0KORJ1()                  (Params: none)
   Called by: DISPLAY.SPR
       Calls: EDDISPLAY                 (procedure in KB.SPR)
Contains: _QSF0KORMJ()                  (Params: none)
   Called by: DISPLAY.SPR
       Calls: EDDISPLAY                 (procedure in KB.SPR)
Contains: _QSF0KORZX()                  (Params: none)
   Called by: DISPLAY.SPR
```

```
Contains: _QSF0KOS30()                  (Params: none)
   Called by: DISPLAY.SPR
Contains: _QSF0KOS6A()                  (Params: none)
   Called by: DISPLAY.SPR
       Calls: EDDISPLAY                 (procedure in KB.SPR)
```

8. DISEASE.SPR

```
Contains: _QSF0KOU0D()                  (Params: none)
   Called by: DISEASE.SPR
Contains: _QSF0KOU3P()                  (Params: none)
   Called by: DISEASE.SPR
Contains: _QSF0KOUMI()                  (Params: none)
   Called by: DISEASE.SPR
Contains: _QSF0KOUQK()                  (Params: none)
   Called by: DISEASE.SPR
```

9. ACTION.SPR

```
Contains: _QSF0KOX5V()                  (Params: none)
   Called by: ACTION.SPR
       Calls: EDACT                     (procedure in ACTION.SPR)
Contains: _QSF0KOX9D()                  (Params: none)
   Called by: ACTION.SPR
       Calls: EDACT                     (procedure in ACTION.SPR)
Contains: _QSF0KOXVA()                  (Params: none)
   Called by: ACTION.SPR
Contains: _QSF0KOY5O()                  (Params: none)
   Called by: ACTION.SPR
       Calls: EDACT                     (procedure in ACTION.SPR)
Contains: _QSF0KOY8A()                  (Params: none)
   Called by: ACTION.SPR
       Calls: CLAUSE.SPR
       Calls: SETACT                    (procedure in KBEDIT.SPR)
Contains: _QSF0KOYCP()                  (Params: none)
   Called by: ACTION.SPR
       Calls: EDACT                     (procedure in ACTION.SPR)
Contains: EDACT                         (Params: none)
   Called by: _QSF0KOX5V()              (function in ACTION.SPR)
   Called by: _QSF0KOX9D()              (function in ACTION.SPR)
   Called by: _QSF0KOY5O()              (function in ACTION.SPR)
   Called by: _QSF0KOYCP()              (function in ACTION.SPR)
       Calls: CLAUSE.SPR
       Calls: SETACT                    (procedure in KBEDIT.SPR)
Contains: ADDNEWACT()                   (Params: none)
   Called by: ACTION.SPR
       Calls: CLAUSE.SPR
       Calls: SETACT                    (procedure in KBEDIT.SPR)
```

10. ACTELSE.SPR

```
Contains: _QSF0KP07D()                  (Params: none)
   Called by: ACTELSE.SPR
       Calls: EDELSE                    (procedure in ACTELSE.SPR)
Contains: _QSF0KP0AR()                  (Params: none)
   Called by: ACTELSE.SPR
       Calls: EDELSE                    (procedure in ACTELSE.SPR)
```

```
Contains: _QSF0KP0S3()                  (Params: none)
   Called by: ACTELSE.SPR
Contains: _QSF0KP0W3()                  (Params: none)
   Called by: ACTELSE.SPR
       Calls: EDELSE                    (procedure in ACTELSE.SPR)
Contains: _QSF0KP0YP()                  (Params: none)
   Called by: ACTELSE.SPR
       Calls: ADDNEWELSE()              (function in ACTELSE.SPR)
       Calls: CLAUSE.SPR
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
Contains: _QSF0KP13M()                  (Params: none)
   Called by: ACTELSE.SPR
       Calls: EDELSE                    (procedure in ACTELSE.SPR)
Contains: EDELSE                        (Params: none)
   Called by: _QSF0KP07D()              (function in ACTELSE.SPR)
   Called by: _QSF0KP0AR()              (function in ACTELSE.SPR)
   Called by: _QSF0KP0W3()              (function in ACTELSE.SPR)
   Called by: _QSF0KP13M()              (function in ACTELSE.SPR)
       Calls: CLAUSE.SPR
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
Contains: ADDELSE                       (Params: none)
       Calls: CLAUSE.SPR
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
Contains: ADDNEWELSE()                  (Params: none)
   Called by: ACTELSE.SPR
   Called by: _QSF0KP0YP()              (function in ACTELSE.SPR)
       Calls: CLAUSE.SPR
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
```

11. CLAUSE.SPR

```
Contains: _QSF0KP3SZ()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP3X2()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP435()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP472()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP5HS()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP5LZ()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP5Q1()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: _QSF0KP5UC()                  (Params: none)
   Called by: CLAUSE.SPR
Contains: INSNEWID()                    (Params: MDATA, MITEM)
   Called by: CLAUSE.SPR
       Calls: DP.PRG
Contains: SETOBJECT()                   (Params: M.OBJECT, MTYPE)
   Called by: CLAUSE.SPR
```

12. RULE.SPR

```
Contains: _QSF0KP97Y()                  (Params: none)
   Called by: RULE.SPR
```

```
Contains: _QSF0KPA3B()                  (Params: none)
   Called by: RULE.SPR
Contains: _QSF0KPA7J()                  (Params: none)
   Called by: RULE.SPR
       Calls: YESNO.PRG
```

13. TERM.SPR

```
Contains: _QSF0KPC3R()                  (Params: none)
   Called by: TERM.SPR
       Calls: EDPREM                    (procedure in TERM.SPR)
Contains: _QSF0KPC6D()                  (Params: none)
   Called by: TERM.SPR
       Calls: EDPREM                    (procedure in TERM.SPR)
Contains: _QSF0KPCPM()                  (Params: none)
   Called by: TERM.SPR
Contains: _QSF0KPCU9()                  (Params: none)
   Called by: TERM.SPR
       Calls: SETJOIN()                 (function in TERM.SPR)
Contains: _QSF0KPCWU()                  (Params: none)
   Called by: TERM.SPR
       Calls: SETJOIN()                 (function in TERM.SPR)
Contains: _QSF0KPCZG()                  (Params: none)
   Called by: TERM.SPR
       Calls: EDPREM                    (procedure in TERM.SPR)
Contains: _QSF0KPD1Z()                  (Params: none)
   Called by: TERM.SPR
       Calls: ADDFACT()                 (function in TERM.SPR)
Contains: _QSF0KPD51()                  (Params: none)
   Called by: TERM.SPR
       Calls: EDPREM                    (procedure in TERM.SPR)
Contains: EDPREM                        (Params: none)
   Called by: _QSF0KPC3R()              (function in TERM.SPR)
   Called by: _QSF0KPC6D()              (function in TERM.SPR)
   Called by: _QSF0KPCZG()              (function in TERM.SPR)
   Called by: _QSF0KPD51()              (function in TERM.SPR)
       Calls: CLAUSE.SPR
       Calls: SETPREM                   (procedure in KBEDIT.SPR)
Contains: SETJOIN()                     (Params: JOINOP)
   Called by: _QSF0KPCU9()              (function in TERM.SPR)
   Called by: _QSF0KPCWU()              (function in TERM.SPR)
       Calls: SETPREM                   (procedure in KBEDIT.SPR)
Contains: LASTFACT()                    (Params: none)
Contains: ADDFACT()                     (Params: none)
   Called by: TERM.SPR
   Called by: _QSF0KPD1Z()              (function in TERM.SPR)
       Calls: CLAUSE.SPR
       Calls: SETPREM                   (procedure in KBEDIT.SPR)
```

14. OBJECT.SPR

```
Contains: _QSF0KPFR1()                  (Params: none)
   Called by: OBJECT.SPR
Contains: _QSF0KPFV8()                  (Params: none)
   Called by: OBJECT.SPR
       Calls: GETOBJ()                  (function in OBJECT.SPR)
Contains: _QSF0KPFYY()                  (Params: none)
```

```
   Called by: OBJECT.SPR
Contains: _QSF0KPG47()                  (Params: none)
   Called by: OBJECT.SPR
       Calls: CREATARRAY()              (function in OBJECT.SPR)
       Calls: SELITEM()                 (function in OBJECT.SPR)
Contains: _QSF0KPGOJ()                  (Params: none)
   Called by: OBJECT.SPR
Contains: _QSF0KPGSR()                  (Params: none)
   Called by: OBJECT.SPR
       Calls: GETOBJ()                  (function in OBJECT.SPR)
Contains: _QSF0KPGVZ()                  (Params: none)
   Called by: OBJECT.SPR
Contains: _QSF0KPH17()                  (Params: none)
   Called by: OBJECT.SPR
       Calls: CREATARRAY()              (function in OBJECT.SPR)
       Calls: SELITEM()                 (function in OBJECT.SPR)
Contains: CREATARRAY()                  (Params: none)
   Called by: _QSF0KPG47()              (function in OBJECT.SPR)
   Called by: _QSF0KPH17()              (function in OBJECT.SPR)
   Called by: GETOBJ()                  (function in OBJECT.SPR)
Contains: DATAINPUT()                   (Params: none)
       Calls: DISEASE.SPR
       Calls: DICT.SPR
Contains: GETOBJ()                      (Params: M.NAME)
   Called by: _QSF0KPFV8()              (function in OBJECT.SPR)
   Called by: _QSF0KPGSR()              (function in OBJECT.SPR)
       Calls: CREATARRAY()              (function in OBJECT.SPR)
       Calls: SELITEM()                 (function in OBJECT.SPR)
Contains: SELITEM()                     (Params: none)
   Called by: _QSF0KPG47()              (function in OBJECT.SPR)
   Called by: _QSF0KPH17()              (function in OBJECT.SPR)
   Called by: GETOBJ()                  (function in OBJECT.SPR)
       Calls: OBLIST.SPR
       Calls: DISEASE.SPR
       Calls: DICT.SPR
```

15. ENUM.SPR

```
Contains: _QSF0KPJAM()                  (Params: none)
   Called by: ENUM.SPR
Contains: _QSF0KPJF7()                  (Params: none)
   Called by: ENUM.SPR
Contains: _QSF0KPJJK()                  (Params: none)
   Called by: ENUM.SPR
Contains: _QSF0KPJYY()                  (Params: none)
   Called by: ENUM.SPR
Contains: _QSF0KPK3S()                  (Params: none)
   Called by: ENUM.SPR
Contains: _QSF0KPK8E()                  (Params: none)
   Called by: ENUM.SPR
```

16. RESTORE.SPR

```
Contains: _QSF0KPMPZ()                  (Params: none)
   Called by: RESTORE.SPR
Contains: _QSF0KPMS0()                  (Params: none)
   Called by: RESTORE.SPR
```

```
        Calls: YESNO.PRG
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPMXI()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: PATHSTRING              (procedure in BACKUP.SPR)
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPN0J()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: PKUNZIP                 (procedure in RESTORE.SPR)
Contains: _QSF0KPN3T()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPN7E()                 (Params: none)
   Called by: RESTORE.SPR
Contains: _QSF0KPNY5()                 (Params: none)
   Called by: RESTORE.SPR
Contains: _QSF0KPO0A()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: YESNO.PRG
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPO5T()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: PATHSTRING              (procedure in BACKUP.SPR)
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPO8T()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: PKUNZIP                 (procedure in RESTORE.SPR)
Contains: _QSF0KPOBY()                 (Params: none)
   Called by: RESTORE.SPR
        Calls: NEWPATHPOP              (procedure in BACKUP.SPR)
        Calls: NEWFILEPOP              (procedure in BACKUP.SPR)
Contains: _QSF0KPOFG()                 (Params: none)
   Called by: RESTORE.SPR
Contains: NEWDRIVEPOP                  (Params: none)
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
Contains: NEWPATHPOP                   (Params: none)
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
   Called by: _QSF0KO98Q()             (function in BACKUP.SPR)
   Called by: _QSF0KO9EC()             (function in BACKUP.SPR)
   Called by: _QSF0KO9LV()             (function in BACKUP.SPR)
   Called by: _QSF0KOAID()             (function in BACKUP.SPR)
   Called by: _QSF0KOANU()             (function in BACKUP.SPR)
   Called by: _QSF0KOAUY()             (function in BACKUP.SPR)
   Called by: _QSF0KPMS0()             (function in RESTORE.SPR)
   Called by: _QSF0KPMXI()             (function in RESTORE.SPR)
   Called by: _QSF0KPN3T()             (function in RESTORE.SPR)
   Called by: _QSF0KPO0A()             (function in RESTORE.SPR)
   Called by: _QSF0KPO5T()             (function in RESTORE.SPR)
   Called by: _QSF0KPOBY()             (function in RESTORE.SPR)
Contains: NEWFILEPOP                   (Params: none)
```

```
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
   Called by: _QSF0KO98Q()          (function in BACKUP.SPR)
   Called by: _QSF0KO9EC()          (function in BACKUP.SPR)
   Called by: _QSF0KO9LV()          (function in BACKUP.SPR)
   Called by: _QSF0KOAID()          (function in BACKUP.SPR)
   Called by: _QSF0KOANU()          (function in BACKUP.SPR)
   Called by: _QSF0KOAUY()          (function in BACKUP.SPR)
   Called by: _QSF0KPMS0()          (function in RESTORE.SPR)
   Called by: _QSF0KPMXI()          (function in RESTORE.SPR)
   Called by: _QSF0KPN3T()          (function in RESTORE.SPR)
   Called by: _QSF0KPO0A()          (function in RESTORE.SPR)
   Called by: _QSF0KPO5T()          (function in RESTORE.SPR)
   Called by: _QSF0KPOBY()          (function in RESTORE.SPR)
Contains: PATHSTRING                (Params: none)
   Called by: _QSF0KO9EC()          (function in BACKUP.SPR)
   Called by: _QSF0KOANU()          (function in BACKUP.SPR)
   Called by: PKZIP                 (procedure in BACKUP.SPR)
   Called by: _QSF0KPMXI()          (function in RESTORE.SPR)
   Called by: _QSF0KPO5T()          (function in RESTORE.SPR)
   Called by: PKUNZIP               (procedure in RESTORE.SPR)
Contains: PKUNZIP                   (Params: none)
   Called by: _QSF0KPN0J()          (function in RESTORE.SPR)
   Called by: _QSF0KPO8T()          (function in RESTORE.SPR)
       Calls: PATHSTRING            (procedure in BACKUP.SPR)
```

17. DD.SPR

```
Contains: VALIDOBJ()                (Params: none)
   Called by: _QSF0KON7E()          (function in QUAL.SPR)
   Called by: _QSF0KPT2I()          (function in DD.SPR)
       Calls: ERRMSG.PRG
Contains: QUALDEL()                 (Params: none)
   Called by: _QSF0KON7E()          (function in QUAL.SPR)
   Called by: _QSF0KPT2I()          (function in DD.SPR)
       Calls: POPUPSHOW()           (function in KBINIT.PRG)
       Calls: POPUPHIDE()           (function in KBINIT.PRG)
       Calls: SETQUALS              (procedure in KB.SPR)
Contains: RESETDATA()               (Params: none)
   Called by: _QSF0KPT2I()          (function in DD.SPR)
Contains: _QSF0KPT2I()              (Params: none)
   Called by: DD.SPR
       Calls: DDEDIT.SPR
       Calls: VALIDOBJ()            (function in QUAL.SPR)
       Calls: QUALDEL()             (function in QUAL.SPR)
       Calls: RESETDATA()           (function in DD.SPR)
Contains: _QSF0KPT7D()              (Params: none)
   Called by: DD.SPR
       Calls: DDEDIT.SPR
```

18. PLAN.SPR

```
Contains: _QSF0KPV8R()              (Params: none)
   Called by: PLAN.SPR
       Calls: OBJECT.SPR
       Calls: SETGOALS              (procedure in KB.SPR)
```

```
Contains: _QSF0KPVCR()                  (Params: none)
   Called by: PLAN.SPR
       Calls: EDGOAL                    (procedure in KB.SPR)
Contains: _QSF0KPVQB()                  (Params: none)
   Called by: PLAN.SPR
       Calls: OBJECT.SPR
       Calls: SETGOALS                  (procedure in KB.SPR)
Contains: _QSF0KPVW8()                  (Params: none)
   Called by: PLAN.SPR
       Calls: EDGOAL                    (procedure in KB.SPR)
```

19. DICT.SPR

```
Contains: EDENUM                        (Params: none)
   Called by: _QSF0KPY8W()              (function in DICT.SPR)
   Called by: _QSF0KQ6T0()              (function in DDEDIT.SPR)
       Calls: ENUM.SPR
       Calls: SETENUM                   (procedure in DICT.SPR)
       Calls: DDENUM.SPR
Contains: SETENUM                       (Params: none)
   Called by: DICT.SPR
   Called by: DDEDIT.SPR
   Called by: EDENUM                    (procedure in DICT.SPR)
   Called by: _QSF0KQ6M1()              (function in DDEDIT.SPR)
Contains: SETVAL                        (Params: none)
   Called by: _QSF0KQ6M1()              (function in DDEDIT.SPR)
Contains: _QSF0KPY4X()                  (Params: none)
   Called by: DICT.SPR
Contains: _QSF0KPY8W()                  (Params: none)
   Called by: DICT.SPR
       Calls: EDENUM                    (procedure in DICT.SPR)
Contains: _QSF0KPYF6()                  (Params: none)
   Called by: DICT.SPR
Contains: _QSF0KPYJ0()                  (Params: none)
   Called by: DICT.SPR
```

20. KBEDIT.SPR

```
Contains: SETPREM                       (Params: none)
   Called by: KBEDIT.SPR
   Called by: _QSF0KOH4Z()              (function in KB.SPR)
   Called by: ADDFACT()                 (function in TERM.SPR)
   Called by: EDPREM                    (procedure in TERM.SPR)
   Called by: SETJOIN()                 (function in TERM.SPR)
   Called by: _QSF0KQ0SA()              (function in KBEDIT.SPR)
   Called by: _QSF0KQ14Z()              (function in KBEDIT.SPR)
       Calls: SEMPTY()                  (function in KBEDIT.SPR)
       Calls: HLINK()                   (function in KBEDIT.SPR)
       Calls: VLINK()                   (function in KBEDIT.SPR)
Contains: VLINK()                       (Params: FROM, TO)
   Called by: SETPREM                   (procedure in KBEDIT.SPR)
Contains: HLINK()                       (Params: S)
   Called by: SETPREM                   (procedure in KBEDIT.SPR)
Contains: PUSH()                        (Params: N)
Contains: POP()                         (Params: none)
Contains: SEMPTY()                      (Params: none)
   Called by: SETPREM                   (procedure in KBEDIT.SPR)
```

```
Contains: SETACT                        (Params: none)
   Called by: KBEDIT.SPR
   Called by: _QSF0KOH4Z()              (function in KB.SPR)
   Called by: ADDNEWACT()               (function in ACTION.SPR)
   Called by: _QSF0KOY8A()              (function in ACTION.SPR)
   Called by: EDACT                     (procedure in ACTION.SPR)
   Called by: _QSF0KQ14Z()              (function in KBEDIT.SPR)
Contains: SETELSE                       (Params: none)
   Called by: KBEDIT.SPR
   Called by: ADDNEWELSE()              (function in ACTELSE.SPR)
   Called by: _QSF0KP0YP()              (function in ACTELSE.SPR)
   Called by: EDELSE                    (procedure in ACTELSE.SPR)
   Called by: ADDELSE                   (procedure in ACTELSE.SPR)
   Called by: _QSF0KQ10R()              (function in KBEDIT.SPR)
   Called by: _QSF0KQ14Z()              (function in KBEDIT.SPR)
Contains: SETACTION()                   (Params: none)
   Called by: KBEDIT.SPR
   Called by: _QSF0KQ14Z()              (function in KBEDIT.SPR)
Contains: _QSF0KQ0SA()                  (Params: none)
   Called by: KBEDIT.SPR
       Calls: TERM.SPR
       Calls: SETPREM                   (procedure in KBEDIT.SPR)
Contains: _QSF0KQ0WK()                  (Params: none)
   Called by: KBEDIT.SPR
       Calls: ACTION.SPR
Contains: _QSF0KQ10R()                  (Params: none)
   Called by: KBEDIT.SPR
       Calls: ACTELSE.SPR
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
Contains: _QSF0KQ14Z()                  (Params: none)
   Called by: KBEDIT.SPR
       Calls: RULE.SPR
       Calls: SETPREM                   (procedure in KBEDIT.SPR)
       Calls: SETACT                    (procedure in KBEDIT.SPR)
       Calls: SETELSE                   (procedure in KBEDIT.SPR)
       Calls: SETACTION()               (function in KBEDIT.SPR)
```

21. OBLIST.SPR

```
Contains: _QSF0KQ4M3()                  (Params: none)
   Called by: OBLIST.SPR
```

22. DDEDIT.SPR

```
Contains: SETRULES()                    (Params: none)
   Called by: DDEDIT.SPR
       Calls: AREANAME()                (function in DDEDIT.SPR)
Contains: EDENUM                        (Params: none)
   Called by: _QSF0KPY8W()              (function in DICT.SPR)
   Called by: _QSF0KQ6T0()              (function in DDEDIT.SPR)
       Calls: ENUM.SPR
       Calls: SETENUM                   (procedure in DICT.SPR)
       Calls: DDENUM.SPR
Contains: SETENUM                       (Params: none)
   Called by: DICT.SPR
   Called by: DDEDIT.SPR
   Called by: EDENUM                    (procedure in DICT.SPR)
```

```
   Called by: _QSF0KQ6M1()              (function in DDEDIT.SPR)
Contains: SETVAL                        (Params: none)
   Called by: _QSF0KQ6M1()              (function in DDEDIT.SPR)
Contains: CLEANENUM()                   (Params: none)
   Called by: _QSF0KQ6YY()              (function in DDEDIT.SPR)
Contains: AREANAME()                    (Params: MID)
   Called by: SETRULES()                (function in DDEDIT.SPR)
Contains: _QSF0KQ6M1()                  (Params: none)
   Called by: DDEDIT.SPR
       Calls: SETVAL                    (procedure in DICT.SPR)
       Calls: SETENUM                   (procedure in DICT.SPR)
       Calls: DDENUM.SPR
Contains: _QSF0KQ6T0()                  (Params: none)
   Called by: DDEDIT.SPR
       Calls: EDENUM                    (procedure in DICT.SPR)
Contains: _QSF0KQ6YY()                  (Params: none)
   Called by: DDEDIT.SPR
       Calls: CLEANENUM()               (function in DDEDIT.SPR)
Contains: _QSF0KQ74F()                  (Params: none)
   Called by: DDEDIT.SPR
```

23. DDENUM.SPR

```
Contains: _QSF0KQ9MB()                  (Params: none)
   Called by: DDENUM.SPR
Contains: _QSF0KQ9QC()                  (Params: none)
   Called by: DDENUM.SPR
```

24. BACKUP.SPR

```
Contains: _QSF0KO96C()                  (Params: none)
   Called by: BACKUP.SPR
Contains: _QSF0KO98Q()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: YESNO.PRG
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
Contains: _QSF0KO9EC()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: PATHSTRING                (procedure in BACKUP.SPR)
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
Contains: _QSF0KO9HK()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: YESNO.PRG
       Calls: PKZIP                     (procedure in BACKUP.SPR)
Contains: _QSF0KO9LV()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
Contains: _QSF0KOAG9()                  (Params: none)
   Called by: BACKUP.SPR
Contains: _QSF0KOAID()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: YESNO.PRG
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
```

```
Contains: _QSF0KOANU()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: PATHSTRING                (procedure in BACKUP.SPR)
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
Contains: _QSF0KOAQV()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: YESNO.PRG
       Calls: PKZIP                     (procedure in BACKUP.SPR)
Contains: _QSF0KOAUY()                  (Params: none)
   Called by: BACKUP.SPR
       Calls: NEWPATHPOP                (procedure in BACKUP.SPR)
       Calls: NEWFILEPOP                (procedure in BACKUP.SPR)
Contains: NEWDRIVEPOP                   (Params: none)
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
Contains: NEWPATHPOP                    (Params: none)
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
   Called by: _QS*                      (function in BACKUP.SPR)
Contains: NEWFILEPOP                    (Params: none)
   Called by: BACKUP.SPR
   Called by: RESTORE.SPR
   Called by: _QSF0KO98Q()              (function in BACKUP.SPR)
   Called by: _QSF0KOAUY()              (function in BACKUP.SPR)
   Called by: _QSF0KPOBY()              (function in RESTORE.SPR)
Contains: PATHSTRING                    (Params: none)
   Called by: _QSF0KO9EC()              (function in BACKUP.SPR)
   Called by: _QSF0KOANU()              (function in BACKUP.SPR)
   Called by: PKZIP                     (procedure in BACKUP.SPR)
   Called by: _QSF0KPMXI()              (function in RESTORE.SPR)
   Called by: _QSF0KPO5T()              (function in RESTORE.SPR)
   Called by: PKUNZIP                   (procedure RESTORE.SPR)
Contains: PKZIP                         (Params: none)
   Called by: _QSF0KO9HK()              (function in BACKUP.SPR)
   Called by: _QSF0KOAQV()              (function in BACKUP.SPR)
       Calls: PATHSTRING                (procedure in BACKUP.SPR)
```

Section VII. Program Source Code

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\KBINIT.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 08/05/94 at 14:05:34
  8:  *:
  9:  *:  Procs & Fncts: SETPATH()
 10:  *:                : MYHANDLER()
 11:  *:                :  QUIT()
 12:  *:                : POPUPSHOW()
 13:  *:                : POPUPHIDE()
 14:  *:
 15:  *:          Calls: SETPATH()          (function in KBINIT.PRG)
 16:  *:                : KBMENU.MPR
 17:  *:                : MYHANDLER()       (function in KBINIT.PRG)
 18:  *:
 19:  *:      Documented 15:00:46                                FoxDoc versio
=> n 3.00a
 20:  *:*************************************************************************
=> *******
 21:      ************************************************************************
=> *
 22:      *
 23:      *    Initialize system init_app()
 24:      *
 25:      ************************************************************************
=> *
 26:      CLOSE ALL
 27:      CLEAR WINDOWS ALL
 28:      CLEAR ALL
 29:      PUBLIC m.pass,savetalk,savesc        && Initialize public variables
 30:      PUBLIC dropdead, mproc
 31:      PUBLIC m.new, m.home, m.data
 32:
 33:     IF SET('TALK') = 'ON'     && TALK handled as a special case.
 34:        SET TALK OFF          && Turn TALK OFF
 35:        savetalk = 'ON'       && TALK was ON, save the setting
 36:     ELSE                      && TALK is OFF
 37:        savetalk = 'OFF'      && TALK was OFF, save the setting
 38:     ENDIF
 39:
 40:     IF SET('ESCAPE') = 'ON'  && ESCAPE handled as a special case.
 41:        SET ESCAPE OFF       && Turn ESCAPE OFF
 42:        savesc = 'ON'        && ESCAPE was ON, save the setting
 43:     ELSE                     && ESCAPE is OFF
 44:        savesc = 'OFF'       && ESCAPE was OFF, save the setting
 45:     ENDIF
 46:
 47:      && End of init_app()
 48:      m.home = SYS(2003)
 49:      m.data = GETENV("KBDATA")
 50:     IF !EMPTY(m.data)
 51:        DO setpath WITH m.data
 52:     ENDIF
 53:      m.bak = m.data + "\bak\"
 54:      m.new = m.data + "\new\"
 55:
 56:      *--------------------------------------------
 57:      * Backup request.
 58:      *--------------------------------------------
 59:      *m.do = yesno("Do you want to Backup database?","YES","NO")
 60:      *IF m.do
 61:      *   do backup
 62:      *ENDIF
 63:
 64:      *setup help file
 65:      SET HELP TO kbhelp.dbf
 66:      ON KEY LABEL f1 HELP
 67:
 68:      *--------------------------------------------
 69:      * Install menu
 70:      *--------------------------------------------
 71:      dropdead = .F.
 72:      PUSH MENU _msysmenu                  && HLL
 73:      DO kbmenu.mpr         && Launch application menu
 74:      READ VALID myhandler()
 75:
 76:
 77:      POP MENU _msysmenu           && HLL
 78:
 79:      *--------------------------------------------
 80:      * Restore request.
 81:      *--------------------------------------------
 82:      CLEAR WINDOWS ALL
 83:      CLOSE DATABASES
 84:      SET TOPIC TO "restore"
 85:      *m.do = yesno("Do you want to Restore?","YES","NO")
 86:      *IF m.do
 87:      *   do restore
 88:      *ENDIF
 89:      SET HELP TO
 90:
 91:      ************************************************************************
=> *
 92:      *     DONE
 93:      ************************************************************************
=> *
 94:      CLEAR WINDOWS ALL
 95:      CLOSE ALL
 96: <==RETURN
 97:
 98:  *!*************************************************************************
=> *******
 99:  *!
100:  *!       Function: MYHANDLER
101:  *!
102:  *!      Called by: KBINIT.PRG
103:  *!
104:  *!*************************************************************************
=> *******
105:      FUNCTION myhandler
106:     IF TYPE("dropdead") = "U"
107:        dropdead = .F.
108:     ENDIF
109: <==RETURN dropdead
110:
111:      *-------------------------------------------------------
112:      *
113:      * Quit the system
114:      *
115:      *-------------------------------------------------------
116:
117:  *!*************************************************************************
=> *******
118:  *!
119:  *!       Function: _QUIT
120:  *!
121:  *!      Called by: KBMENU.MPR
```

```
122:  *!
123:  *!          Calls: ERRMSG.PRG
124:  *!
125:  *!*************************************************************************
=> *******
126:      FUNCTION _quit
127:     IF EMPTY(_WOUTPUT())
128:        dropdead = .T.
129:        CLEAR READ ALL
130:     ELSE
131:        =errmsg("Close windows before quitting",1)
132:     ENDIF
133: <---RETURN
134:
135:     *-------------------------------------------------------
136:     *
137:     * display popup notice
138:     *
139:     *-------------------------------------------------------
140:
141:  *!*************************************************************************
=> *******
142:  *!
143:  *!        Function: POPUPSHOW
144:  *!
145:  *!      Called by: QUALDEL()          (function in QUAL.SPR)
146:  *!               : _QSFOKQC6T()       (function in KBLOAD.SPR)
147:  *!
148:  *!*************************************************************************
=> *******
149:      FUNCTION popupshow
150:      PARAMETERS errstr
151:     IF NOT WEXIST("w_popnote")
152:        DEFINE WINDOW w_popnote ;
153:           FROM INT((SROW()-8)/2),INT((SCOL()-36)/2) ;
154:           TO INT((SROW()-8)/2)+7,INT((SCOL()-36)/2)+35 ;
155:           TITLE "One moment" ;
156:           FLOAT ;
157:           CLOSE ;
158:           SHADOW ;
159:           DOUBLE ;
160:           COLOR SCHEME 1
161:     ENDIF
162:     IF WVISIBLE("w_popnote")
163:        ACTIVATE WINDOW w_popnote SAME
164:     ELSE
165:        ACTIVATE WINDOW w_popnote NOSHOW
166:     ENDIF
167:      @ 1,1 SAY errstr SIZE 3,31
168:
169:     IF NOT WVISIBLE("w_popnote")
170:        ACTIVATE WINDOW w_popnote
171:     ENDIF
172: <---RETURN ""
173:
174:  *!*************************************************************************
=> *******
175:  *!
176:  *!        Function: POPUPHIDE
177:  *!
178:  *!      Called by: QUALDEL()          (function in QUAL.SPR)
179:  *!               : _QSFOKQC6T()       (function in KBLOAD.SPR)
180:  *!
181:  *!*************************************************************************
=> *******
182:      FUNCTION popuphide
183:      RELEASE WINDOW w_popnote
184: <---RETURN
185:
186:     *-------------------------------------------
187:     * Quit out the system
188:     *-------------------------------------------
189:     CLOSE DATABASES
190:     *set default to (m.home)
191:
192:     *-----------------------------------------------------
193:     * Creat foxpro path
194:     *-----------------------------------------------------
195:  *!*************************************************************************
=> *******
196:  *!
197:  *!        Function: SETPATH
198:  *!
199:  *!      Called by: KBINIT.PRG
200:  *!
201:  *!*************************************************************************
=> *******
202:      FUNCTION setpath
203:      PARAMETER newpath
204:      PRIVATE mnewpath
205:     IF !EMPTY(newpath)
206:        mnewpath = FULLPATH(newpath) + ";"
207:        mcurpath = IIF(EMPTY(SET("PATH")),FULLPATH(CURDIR())+";",SET("PATH
=> "))
208:        IF NOT mcurpath $ mnewpath
209:           mcurpath = mcurpath + mnewpath      && add sams subdirectory to
=>  path
210:        ENDIF
211:        SET PATH TO (mcurpath)
212:     ENDIF
213: <---RETURN .T.
214:
215:
216:
217:
218:  *:  EOF: KBINIT.act
```

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\YESNO.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 08/05/94 at 11:38:50
  8:  *:
  9:  *:         Set by: _QSFOKO98Q()       (function in BACKUP.SPR)
 10:  *:               : _QSFOKO9HK()       (function in BACKUP.SPR)
 11:  *:               : _QSFOKOAID()       (function in BACKUP.SPR)
 12:  *:               : _QSFOKOAQV()       (function in BACKUP.SPR)
 13:  *:               : _QSFOKPA7J()       (function in RULE.SPR)
 14:  *:               : _QSFOKPMSO()       (function in RESTORE.SPR)
 15:  *:               : _QSFOKPOOA()       (function in RESTORE.SPR)
 16:  *:               : CHECKFILE()        (function in KBLOAD.SPR)
 17:  *:
 18:  *:      Documented 15:00:47                                FoxDoc versio
=> n 3.00a
 19:  *:*************************************************************************
=> *******
 20:      *       ************************************************************
 21:      *       *                                                          *
 22:      *       * 11/26/91              YESNO.PRG                 02:50:49 *
 23:      *       *                                                          *
 24:      *       ************************************************************
 25:      *       *                                                          *
 26:      *       * Adam Green                                               *
 27:      *       *                                                          *
 28:      *       * Copyright (c) 1991 Adam Green Seminars                   *
 29:      *       * One Faneuil Hall                                         *
 30:      *       * Boston, MA  02174                                        *
 31:      *       *                                                          *
 32:      *       * Description:                                             *
 33:      *       * This program was automatically generated by GENSCRN.     *
 34:      *       *                                                          *
 35:      *       ************************************************************
 36:
 37:
 38:      *       ************************************************************
 39:      *       *                                                          *
 40:      *       *              YESNO Setup Code - SECTION 1                *
 41:      *       *                                                          *
 42:      *       ************************************************************
 43:      *
 44:
 45:      #REGION 1
 46:      PARAMETERS MESSAGE, ok, CANCEL
 47:      PRIVATE ALL
 48:      *HLL, If message was empty print a plain window just only need respon
=> se
 49:     IF PARAMETERS() = 0
 50:        * Default to plain prompts
 51:        m.message = ""
 52:     ENDIF
 53:
 54:      * Ok and Cancel are used for prompts in the push buttons
 55:     IF PARAMETERS() = 1
 56:        * Default to normal prompts
 57:        m.ok = "OK"
 58:        m.cancel = "Cancel"
 59:     ENDIF
 60:
 61:      * Truncate any message longer than 50 characters
 62:     IF LEN( m.message ) > 129
 63:        * The message is centered in an @SAY
 64:        m.message = SUBSTR( m.message, 1, 129 )
 65:     ENDIF
 66:
 67:      PUSH KEY CLEAR
 68:
 69:      #REGION 0
 70:      REGIONAL m.currarea, m.talkstat, m.compstat
 71:
 72:     IF SET("TALK") = "ON"
 73:        SET TALK OFF
 74:        m.talkstat = "ON"
 75:     ELSE
 76:        m.talkstat = "OFF"
 77:     ENDIF
 78:      m.compstat = SET("COMPATIBLE")
 79:      SET COMPATIBLE FOXPLUS
 80:
 81:      *       ************************************************************
 82:      *       *                                                          *
 83:      *       *                     Window definitions                   *
 84:      *       *                                                          *
 85:      *       ************************************************************
 86:      *
 87:
 88:     IF NOT WEXIST("yesno")
 89:        DEFINE WINDOW yesno ;
 90:           AT 0,0 ;
 91:           SIZE 7,72 ;
 92:           FONT "MS SANS SERIF", 8 ;
 93:           FLOAT ;
 94:           NOCLOSE ;
 95:           DOUBLE
 96:        MOVE WINDOW yesno CENTER
 97:
 98:     ENDIF
 99:
100:
101:      *       ************************************************************
102:      *       *                                                          *
103:      *       *                   YESNO Screen Layout                    *
104:      *       *                                                          *
105:      *       ************************************************************
106:      *
107:
108:      #REGION 1
109:     IF WVISIBLE("yesno")
110:        ACTIVATE WINDOW yesno SAME
111:     ELSE
112:        ACTIVATE WINDOW yesno NOSHOW
113:     ENDIF
114:      @ 1,2 SAY m.message ;
115:        FONT "MS Sans Serif", 8 ;
116:        SIZE 2,65
117:      @ 4,18 GET m.answer ;
118:        PICTURE "@*HT \!\<&OK;\<&Cancel" ;
119:        SIZE 1.5,11,4 ;
120:        FONT "MS Sans Serif", 8 ;
121:        DEFAULT 1;
122:        STYLE "B"
123:
124:     IF NOT WVISIBLE("yesno")
125:        ACTIVATE WINDOW yesno
126:     ENDIF
127:
```

```
128:     READ CYCLE MODAL
129:
130:     RELEASE WINDOW yesno
131:
132:     #REGION 0
133:    ┌IF m.talkstat = "ON"
134:    │   SET TALK ON
135:    └ENDIF
136:    ┌IF m.compstat = "ON"
137:    │   SET COMPATIBLE ON
138:    └ENDIF
139:
140:
141:     *       ************************************************************
142:     *       *                                                          *
143:     *       *                    YESNO Cleanup Code                    *
144:     *       *                                                          *
145:     *       ************************************************************
146:     *
147:
148:     #REGION 1
149:     POP KEY
150:
151:     * Convert the numeric value of m.Answer from 1|2 to .T.|.F.
152:     * If the user selected OK and didn't exit with Escape
153:    ┌IF m.answer = 1 AND LASTKEY() <> 27
154: <══════RETURN .T.
155:    ├ELSE
156:    │   * Cancel or Escape returns false
157: <══════RETURN .F.
158:    └ENDIF
159:  *:  EOF: YESNO.act
```

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\BACKUP.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 02/15/94 at 13:31:20
  8:  *:
  9:  *:         Set by: KBMENU.MPR
 10:  *:
 11:  *:          Calls: BACKUP.SPR
 12:  *:
 13:  *:      Documented 15:00:47                                  FoxDoc versio
=> n 3.00a
 14:  *:*************************************************************************
=> *******
 15:      *-------------------------------------------------------------------
=> ----
 16:      * Backup data files
 17:      * Programmer: HLL
 18:      *-------------------------------------------------------------------
=> ---
 19:      * PROCEDURE backup
 20:      PUBLIC mscreen
 21:      PRIVATE MESSAGE
 22:      SAVE SCREEN TO mscreen
 23:      MESSAGE = "Backup Knowledge Base database"
 24:      SET TOPIC TO "BACKUP"
 25:      DO backup.spr WITH MESSAGE
 26:      RESTORE SCREEN FROM mscreen
 27:  <==RETURN
 28:
 29:   *:  EOF: BACKUP.act
```

C-A1

```
 1:     *
 2:     *
 3:     *        08/09/94              KBMENU.MPR                09:38:40
 4:     *
 5:     *
 6:     *        Author's Name
 7:     *
 8:     *        Copyright (c) 1994 Company Name
 9:     *        Address
10:     *        City,     Zip
11:     *
12:     *        Description:
13:     *        This program was automatically generated by GENMENU.
14:     *
15:     *
16:     *
17:
18:
19:     *
20:     *                          Menu Definition
21:     *
22:     *
23:     *
24:     *
25:
26:     SET SYSMENU TO
27:
28:     SET SYSMENU AUTOMATIC
29:
30:     DEFINE PAD _qsf0ko6j4 OF _msysmenu PROMPT "System" COLOR SCHEME 3
31:     DEFINE PAD _qsf0ko6je OF _msysmenu PROMPT "Knowledge Base" COLOR SCHE
=> ME 3
32:     DEFINE PAD _qsf0ko6jm OF _msysmenu PROMPT "Exit system" COLOR SCHEME
=> 3
33:     ON PAD _qsf0ko6j4 OF _msysmenu ACTIVATE POPUP SYSTEM
34:     ON PAD _qsf0ko6je OF _msysmenu ACTIVATE POPUP knowledgeb
35:     ON SELECTION PAD _qsf0ko6jm OF _msysmenu DO _quit
36:
37:     DEFINE POPUP SYSTEM MARGIN RELATIVE SHADOW COLOR SCHEME 4
38:     DEFINE BAR _mst_help OF SYSTEM PROMPT "\<Help                  F1"
39:     DEFINE BAR 2 OF SYSTEM PROMPT "\-"
40:     DEFINE BAR 3 OF SYSTEM PROMPT "\<Backup"
41:     DEFINE BAR 4 OF SYSTEM PROMPT "\<Restore"
42:     ON SELECTION BAR 3 OF SYSTEM DO backup
43:     ON SELECTION BAR 4 OF SYSTEM DO RESTORE
44:
45:     DEFINE POPUP knowledgeb MARGIN RELATIVE SHADOW COLOR SCHEME 4
46:     DEFINE BAR 1 OF knowledgeb PROMPT "Knowledge Base Area"
47:     DEFINE BAR 2 OF knowledgeb PROMPT "Question - Data Dictionary"
48:     ON SELECTION BAR 1 OF knowledgeb DO kb.spr
49:     ON SELECTION BAR 2 OF knowledgeb DO dd.spr
50: *:  EOF: KBMENU.ac2
```

```
*
*       ┌──────────────────────────────────────────────────┐
*       │ 08/09/94          BACKUP.SPR             09:38:42 │
*       │                                                  │
*       │ Author's Name                                    │
*       │                                                  │
*       │ Copyright (c) 1994 Company Name                  │
*       │ Address                                          │
*       │ City,     Zip                                    │
*       │                                                  │
*       │ Description:                                     │
*       │ This program was automatically generated by GENSCRN.
*       │                                                  │
*       └──────────────────────────────────────────────────┘
*

PARAMETERS MESSAGE, wildcard, filename
DO CASE
CASE _WINDOWS

    *   ┌──────────────────────────────────────────────────┐
    *   │                                                  │
    *   │          BACKUP/Windows Setup Code - SECTION 1   │
    *   │                                                  │
    *   └──────────────────────────────────────────────────┘
    *

    #REGION 1
    PRIVATE mfanme, mpath, olddrive, predrive, oldpath, maction, mdrive, mfiles
    DO CASE
    CASE PARAMETERS() = 0
        m.message = ""
        m.wildcard = "*.ZIP"
        mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
    CASE PARAMETERS() = 1 OR EMPTY(m.wildcard)
        m.wildcard = "*.ZIP"
        mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
    CASE PARAMETERS() = 2 OR EMPTY(m.filename)
        mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
    OTHERWISE
        mfname = filename
    ENDCASE


    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
    ELSE
        m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    m.rborder = SET("READBORDER")
    SET readborder ON

    *   ┌──────────────────────────────────────────────────┐
    *   │                                                  │
    *   │          Windows Window definitions              │
    *   │                                                  │
    *   └──────────────────────────────────────────────────┘
    *

IF NOT WEXIST("w_backup") ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.PJX" ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.SCX" ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.MNX" ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.PRG" ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.FRX" ;
      OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.QPR"
   DEFINE WINDOW w_backup ;
      AT  0.000, 0.000  ;
      SIZE 22.167,52.500 ;
      FONT "Terminal", 8 ;
      NOFLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      DOUBLE
   MOVE WINDOW w_backup CENTER
ENDIF

    *   ┌──────────────────────────────────────────────────┐
    *   │                                                  │
    *   │          BACKUP/Windows Setup Code - SECTION 2   │
    *   │                                                  │
    *   └──────────────────────────────────────────────────┘
    *

    #REGION 1
    m.olddrive = SET("DEFAULT")
    m.prevdrive = 1
    m.oldpath = CURDIR()
    maction = 1

    DECLARE drivearray[1,1]
    mdrive = 1
    DO newdrivepop

    DECLARE patharray[1,1]
    mpath = 1
    DO newpathpop

    DECLARE filearray[1,1]
    mfiles = 1
    DO newfilepop


    *=closefiles()
```

```
*
*
*       BACKUP/Windows Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("w_backup")
   ACTIVATE WINDOW w_backup SAME
ELSE
   ACTIVATE WINDOW w_backup NOSHOW
ENDIF
@ 5.083,26.375 SAY "To drive: "  ;
   FONT "Terminal", 8
@ 4.917,37.375 GET mdrive ;
   PICTURE "@^" ;
   FROM drivearray ;
   SIZE 1.500,12.500 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   WHEN _qsf0ko96c() ;
   VALID _qsf0ko98q()
@ 8.667,26.125 SAY "Directory:"  ;
   FONT "Terminal", 8
@ 0.917,3.125 SAY MESSAGE ;
   SIZE 1.333,42.500 ;
   FONT "Terminal", 8
@ 8.333,37.500 GET mpath ;
   PICTURE "@^" ;
   FROM patharray ;
   SIZE 1.500,12.500 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0ko9ec()
@ 12.917,26.500 GET maction ;
   PICTURE "@*HT \!\<Backup;\<Cancel" ;
   SIZE 2.083,9.625,2.500 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0ko9hk()
@ 3.750,3.000 GET mfile ;
   PICTURE "@&N" ;
   FROM filearray ;
   SIZE 14.000,21.375 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0ko9lv()
@ 18.667,3.125 SAY "Backup file name: "  ;
   FONT "Terminal", 8
@ 18.583,22.375 GET mfname ;
   SIZE 1.083,27.000 ;
   DEFAULT " " ;
   FONT "Terminal", 8

IF NOT WVISIBLE("w_backup")
   ACTIVATE WINDOW w_backup
ENDIF

READ CYCLE MODAL

RELEASE WINDOW w_backup

#REGION 0

SET readborder &rborder

IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


*
*
*       BACKUP/Windows Cleanup Code
*
*
*

#REGION 1
SET DEFAULT TO (m.olddrive + m.oldpath)
RETURN

****************************************************

*********************

CASE _DOS


*
*
*       BACKUP/MS-DOS Setup Code - SECTION 1
*
*
*

#REGION 1
PRIVATE mfanme, mpath, olddrive, predrive, oldpath, maction, mdriv
=> e, mfiles
DO CASE
CASE PARAMETERS() = 0
   m.message = ""
   m.wildcard = "*.ZIP"
   mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
CASE PARAMETERS() = 1 OR EMPTY(m.wildcard)
   m.wildcard = "*.ZIP"
   mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
CASE PARAMETERS() = 2 OR EMPTY(m.filename)
   mfname = "BU" + STRTRAN(DTOC(DATE()),"/","")
```

```
  ┌OTHERWISE
  │   mfname = filename
  └ENDCASE


  #REGION 0
  REGIONAL m.currarea, m.talkstat, m.compstat

┌IF SET("TALK") = "ON"
│   SET TALK OFF
│   m.talkstat = "ON"
├ELSE
│   m.talkstat = "OFF"
└ENDIF
  m.compstat = SET("COMPATIBLE")
  SET COMPATIBLE FOXPLUS
  *
  *
  *              MS-DOS Window definitions
  *
  *
  *

┌IF NOT WEXIST("w_backup") ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.PJX" ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.SCX" ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.MNX" ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.PRG" ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.FRX" ;
│       OR UPPER(WTITLE("W_BACKUP")) == "W_BACKUP.QPR"
│   DEFINE WINDOW w_backup ;
│       FROM INT((SROW()-18)/2),INT((SCOL()-61)/2) ;
│       TO INT((SROW()-18)/2)+17,INT((SCOL()-61)/2)+60 ;
│       NOFLOAT ;
│       NOCLOSE ;
│       SHADOW ;
│       NOMINIMIZE ;
│       DOUBLE ;
│       COLOR SCHEME 5
└ENDIF


  *
  *
  *              BACKUP/MS-DOS Setup Code - SECTION 2
  *
  *
  *

  #REGION 1
  m.olddrive = SET("DEFAULT")
  m.prevdrive = 1
  m.oldpath = CURDIR()
  maction = 1

  DECLARE drivearray[1,1]
  mdrive = 1
  DO newdrivepop

  DECLARE patharray[1,1]
  mpath = 1
  DO newpathpop

  DECLARE filearray[1,1]
  mfiles = 1
  DO newfilepop


  *=closefiles()






  *
  *
  *              BACKUP/MS-DOS Screen Layout
  *
  *
  *


  #REGION 1
┌IF WVISIBLE("w_backup")
│   ACTIVATE WINDOW w_backup SAME
├ELSE
│   ACTIVATE WINDOW w_backup NOSHOW
└ENDIF
  @ 3,29 SAY "To drive: " ;
     SIZE 1,10, 0
  @ 2,40 GET mdrive ;
     PICTURE "@^" ;
     FROM drivearray ;
     SIZE 3,18 ;
     DEFAULT 1 ;
     WHEN _qsf0koag9() ;
     VALID _qsf0koaid() ;
     COLOR SCHEME 5, 6
  @ 6,29 SAY "Directory:" ;
     SIZE 1,10, 0
  @ 0,1 SAY MESSAGE ;
     SIZE 1,40
  @ 5,40 GET mpath ;
     PICTURE "@^" ;
     FROM patharray ;
     SIZE 3,18 ;
     DEFAULT 1 ;
     VALID _qsf0koanu() ;
     COLOR SCHEME 5, 6
  @ 9,45 GET maction ;
     PICTURE "@*VT \!\<Backup;\<Cancel" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0koaqv()
  @ 2,1 GET mfile ;
     PICTURE "@&N" ;
     FROM filearray ;
```

```
        SIZE 11,26 ;
        DEFAULT 1 ;
        VALID _qsf0koauy() ;
        COLOR SCHEME 6
    @ 14,1 SAY "Backup file name: " ;
        SIZE 1,18, 0
    @ 14,19 GET mfname ;
        SIZE 1,39 ;
        DEFAULT " "

    IF NOT WVISIBLE("w_backup")
        ACTIVATE WINDOW w_backup
    ENDIF

    READ CYCLE MODAL

    RELEASE WINDOW w_backup

    #REGION 0
    IF m.talkstat = "ON"
        SET TALK ON
    ENDIF
    IF m.compstat = "ON"
        SET COMPATIBLE ON
    ENDIF


    *
    *
    *          BACKUP/MS-DOS Cleanup Code
    *
    *
    *

    #REGION 1
    SET DEFAULT TO (m.olddrive + m.oldpath)
    RETURN

    ***********************************************************

    *********************

ENDCASE


*
*
*       _QSF0KO96C          mdrive WHEN
*
*       Function Origin:
*
*       From Platform:      Windows
*       From Screen:        BACKUP,     Record Number:    3
*       Variable:           mdrive
*       Called By:          WHEN Clause
*       Object Type:        Popup
*       Snippet Number:     1
*
*
*
FUNCTION _qsf0ko96c     &&  mdrive WHEN
#REGION 1
m.prevdrive = mdrive

*
*
*       _QSF0KO98Q          mdrive VALID
*
*       Function Origin:
*
*       From Platform:      Windows
*       From Screen:        BACKUP,     Record Number:    3
*       Variable:           mdrive
*       Called By:          VALID Clause
*       Object Type:        Popup
*       Snippet Number:     2
*
*
*
*Switch to the selected drive
FUNCTION _qsf0ko98q     &&  mdrive VALID
#REGION 1
PRIVATE newdrive,mready

*Convert the popup bar number into the matching drive name
m.newdrive = drivearray[mdrive]

IF UPPER(m.newdrive) $ "A:B:"
    mready = yesno("Please insert disk into drive " + m.newdrive, "rea
=> dy","cancel")
ELSE
    mready = .T.
ENDIF
IF mready
    *Go there and reset all the other popups to match
    SET DEFAULT TO (m.newdrive)
    DO newpathpop
    DO newfilepop
ELSE
    mdrive =  m.prevdrive
    m.newdrive = drivearray[mdrive]
ENDIF
SHOW GETS

 *
 *
 *       _QSF0KO9EC          mpath VALID
 *
 *       Function Origin:
 *
 *       From Platform:      Windows
 *       From Screen:        BACKUP,     Record Number:    6
 *       Variable:           mpath
 *       Called By:          VALID Clause
 *       Object Type:        Popup
 *       Snippet Number:     3
 *
 *
 *
 FUNCTION _qsf0ko9ec     &&  mpath VALID
 #REGION 1
 m.newdefault = pathstring()
 SET DEFAULT TO (m.newdefault)
 DO newpathpop
 DO newfilepop
 SHOW GETS
```

```
476:     *
477:     *
478:     *          _QSF0KO9HK            maction VALID
479:     *
480:     *          Function Origin:
481:     *
482:     *          From Platform:        Windows
483:     *          From Screen:          BACKUP,     Record Number:    7
484:     *          Variable:             maction
485:     *          Called By:            VALID Clause
486:     *          Object Type:          Push Button
487:     *          Snippet Number:       4
488:     *
489:     *
490:     *
491:     FUNCTION _qsf0ko9hk     &&  maction VALID
492:     #REGION 1
493:    ┌DO CASE
494:    ├CASE maction = 1
495:    │
496:    │   pkzip = .T.
497:    │  ┌IF FILE(mfname)
498:    │  │   mdel = yesno( mfname + " already exists, overwrite it? ", "Yes"
=>  , "Cancel")
499:    │  │  ┌IF mdel
500:    │  │  │   DELETE FILE &mfname
501:    │  │  ├ELSE
502:    │  │  │   pkzip = .F.
503:    │  │  └ENDIF
504:    │  └ENDIF
505:    │  ┌IF pkzip
506:    │  │   DO pkzip
507:    │  └ENDIF
508:    ├OTHERWISE
509:    └ENDCASE
510:
511:     *
512:     *
513:     *          _QSF0KO9LV            mfile VALID
514:     *
515:     *          Function Origin:
516:     *
517:     *          From Platform:        Windows
518:     *          From Screen:          BACKUP,     Record Number:    8
519:     *          Variable:             mfile
520:     *          Called By:            VALID Clause
521:     *          Object Type:          List
522:     *          Snippet Number:       5
523:     *
524:     *
525:     *
526:     FUNCTION _qsf0ko9lv     &&  mfile VALID
527:     #REGION 1
528:     mnewfile = filearray[mfile,1]
529:    ┌IF "[" $ mnewfile
530:    │   mnewpath = SUBSTR(mnewfile,2,LEN(mnewfile)-2)
531:    │   SET DEFA TO (mnewpath)
532:    │   DO newpathpop
533:    │   DO newfilepop
534:    │   SHOW GETS
535:    ├ELSE
536:    │   mfname = mnewfile
537:    │   SHOW GETS
538:    └ENDIF
539:
540:     *
```

```
541:     *
542:     *          _QSF0KOAG9            mdrive WHEN
543:     *
544:     *          Function Origin:
545:     *
546:     *          From Platform:        MS-DOS
547:     *          From Screen:          BACKUP,     Record Number:    14
548:     *          Variable:             mdrive
549:     *          Called By:            WHEN Clause
550:     *          Object Type:          Popup
551:     *          Snippet Number:       6
552:     *
553:     *
554:     *
555:     FUNCTION _qsf0koag9     &&  mdrive WHEN
556:     #REGION 1
557:     m.prevdrive = mdrive
558:
559:     *
560:     *
561:     *          _QSF0KOAID            mdrive VALID
562:     *
563:     *          Function Origin:
564:     *
565:     *          From Platform:        MS-DOS
566:     *          From Screen:          BACKUP,     Record Number:    14
567:     *          Variable:             mdrive
568:     *          Called By:            VALID Clause
569:     *          Object Type:          Popup
570:     *          Snippet Number:       7
571:     *
572:     *
573:     *
574:     *Switch to the selected drive
575:     FUNCTION _qsf0koaid     &&  mdrive VALID
576:     #REGION 1
577:     PRIVATE newdrive,mready
578:
579:     *Convert the popup bar number into the matching drive name
580:     m.newdrive = drivearray[mdrive]
581:
582:    ┌IF UPPER(m.newdrive) $ "A:B:"
583:    │   mready = yesno("Please insert disk into drive " + m.newdrive, "rea
=> dy","cancel")
584:    ├ELSE
585:    │   mready = .T.
586:    └ENDIF
587:    ┌IF mready
588:    │   *Go there and reset all the other popups to match
589:    │   SET DEFAULT TO (m.newdrive)
590:    │   DO newpathpop
591:    │   DO newfilepop
592:    ├ELSE
593:    │   mdrive =  m.prevdrive
594:    │   m.newdrive = drivearray[mdrive]
595:    └ENDIF
596:     SHOW GETS
597:
598:     *
599:     *
600:     *          _QSF0KOANU            mpath VALID
601:     *
602:     *          Function Origin:
603:     *
604:     *          From Platform:        MS-DOS
605:     *          From Screen:          BACKUP,     Record Number:    17
```

```
606:     *          Variable:            mpath
607:     *          Called By:           VALID Clause
608:     *          Object Type:         Popup
609:     *          Snippet Number:      8
610:     *
611:     *
612:     *
613:     FUNCTION _qsf0koanu     &&  mpath VALID
614:     #REGION 1
615:     m.newdefault = pathstring()
616:     SET DEFAULT TO (m.newdefault)
617:     DO newpathpop
618:     DO newfilepop
619:     SHOW GETS
620:
621:     *
622:     *
623:     *          _QSF0KOAQV           maction VALID
624:     *
625:     *          Function Origin:
626:     *
627:     *          From Platform:       MS-DOS
628:     *          From Screen:         BACKUP,     Record Number:   18
629:     *          Variable:            maction
630:     *          Called By:           VALID Clause
631:     *          Object Type:         Push Button
632:     *          Snippet Number:      9
633:     *
634:     *
635:     *
636:     FUNCTION _qsf0koaqv     &&  maction VALID
637:     #REGION 1
638:     DO CASE
639:     CASE maction = 1
640:
641:        pkzip = .T.
642:        IF FILE(mfname)
643:           mdel = yesno( mfname + " already exists, overwrite it? ", "Yes"
=>, "Cancel")
644:           IF mdel
645:              DELETE FILE &mfname
646:           ELSE
647:              pkzip = .F.
648:           ENDIF
649:        ENDIF
650:        IF pkzip
651:           DO pkzip
652:        ENDIF
653:     OTHERWISE
654:     ENDCASE
655:
656:     *
657:     *
658:     *          _QSF0KOAUY           mfile VALID
659:     *
660:     *          Function Origin:
661:     *
662:     *          From Platform:       MS-DOS
663:     *          From Screen:         BACKUP,     Record Number:   19
664:     *          Variable:            mfile
665:     *          Called By:           VALID Clause
666:     *          Object Type:         List
667:     *          Snippet Number:      10
668:     *
669:     *
670:     *
```

```
671:     FUNCTION _qsf0koauy     &&  mfile VALID
672:     #REGION 1
673:     mnewfile = filearray[mfile,1]
674:     IF "[" $ mnewfile
675:        mnewpath = SUBSTR(mnewfile,2,LEN(mnewfile)-2)
676:        SET DEFA TO (mnewpath)
677:        DO newpathpop
678:        DO newfilepop
679:        SHOW GETS
680:     ELSE
681:        mfname = mnewfile
682:        SHOW GETS
683:     ENDIF
684:
685:
686:     *
687:     *
688:     *          BACKUP/MS-DOS Supporting Procedures and Functions
689:     *
690:     *
691:     *
692:
693:     #REGION 1
694:
695:     *
696:     *
697:     *                    BACKUP Procedure NEWDRIVEPOP
698:     *
699:     *
700:     *
701:
702:
703:     PROCEDURE newdrivepop
704:     DO CASE
705:     CASE _DOS
706:        **********************
707:        * Create a popup with each of the legal drive names
708:        * The popup will be based on an array of the drive names
709:        PRIVATE olddefault, olderror, drivename, ERROR
710:
711:        * We will change drives, so remember where we started
712:        m.olddefault = SET( "DEFAULT" )
713:
714:        * To test for legal drives this program SET DEFAULT TO each drive
715:        * If no error occurs the drive name will be added to an array
716:
717:        * Create the error trap
718:        m.olderror = ON( "ERROR" )
719:        m.error = .F.
720:        ON ERROR m.error = .T.
721:
722:        * Create the array of legal drive names
723:        DECLARE drivearray[ 1 ]
724:        drivearray[ 1 ] = ""
725:
726:        * Loop from A to Z
727:        m.drivename = "A"
728:        FOR i = 1 TO 26
729:
730:           * Try to switch to the drive
731:           SET DEFAULT TO (m.drivename)
732:
733:           * If it worked
734:           IF NOT m.error
735:
736:              * Add the name to the last element in the array
```

```
737:                drivearray[ ALEN( DriveArray ) ] = m.drivename + ":"
738:
739:                * Add another element at the end of the array
740:                DECLARE drivearray[ ALEN( DriveArray ) + 1 ]
741:
742:             ENDIF
743:
744:             * Change to the next letter in the alphabet
745:             m.drivename = CHR( ASC( m.drivename ) + 1 )
746:
747:             * Reset the error trap
748:             m.error = .F.
749:          NEXT
750:
751:          * If the first element is empty, no drives were found
752:          IF EMPTY( drivearray[ 1 ] )
753:             SHOW GET mdrive disabled
754:          ELSE
755:             * Drives were found so cut off the last empty element
756:             * added to the array in the loop
757:             DECLARE drivearray[ ALEN( DriveArray ) - 1 ]
758:          ENDIF
759:
760:          * Reset the error trap and return to home
761:          ON ERROR &olderror
762:          SET DEFAULT TO (m.olddefault)
763:
764:          * Initialize the popup variable to the element in the
765:          * drive array which contains the current drive
766:          mdrive = ASCAN( drivearray, m.olddefault )
767: <--------RETURN
768:
769:          **********
770:       CASE  WINDOWS
771:          **********
772:          * Create a popup with each of the legal drive names
773:          * The popup will be based on an array of the drive names
774:          PRIVATE olddefault, olderror, drivename, ERROR
775:
776:          * We will change drives, so remember where we started
777:          m.olddefault = SET( "DEFAULT" )
778:
779:          * To test for legal drives this program SET DEFAULT TO each drive
780:          * If no error occurs the drive name will be added to an array
781:
782:          * Create the error trap
783:          m.olderror = ON( "ERROR" )
784:          m.error = .F.
785:          ON ERROR m.error = .T.
786:
787:          * Create the array of legal drive names
788:          DECLARE drivearray[ 3 ]
789:          drivearray[ 1 ] = "A"
790:          drivearray[ 2 ] = "B"
791:          drivearray[ 3 ] = ""
792:
793:          * Loop from C to Z
794:          m.drivename = "C"
795:          FOR i = 3 TO 26
796:
797:             * Try to switch to the drive
798:             SET DEFAULT TO (m.drivename)
799:
800:             * If it worked
801:             IF NOT m.error
802:
```



```
803:                * Add the name to the last element in the array
804:                drivearray[ ALEN( DriveArray ) ] = m.drivename + ":"
805:
806:                * Add another element at the end of the array
807:                DECLARE drivearray[ ALEN( DriveArray ) + 1 ]
808:
809:             ENDIF
810:
811:             * Change to the next letter in the alphabet
812:             m.drivename = CHR( ASC( m.drivename ) + 1 )
813:
814:             * Reset the error trap
815:             m.error = .F.
816:          NEXT
817:
818:          * If the first element is empty, no drives were found
819:          IF EMPTY( drivearray[ 1 ] )
820:             SHOW GET mdrive disabled
821:          ELSE
822:             * Drives were found so cut off the last empty element
823:             * added to the array in the loop
824:             DECLARE drivearray[ ALEN( DriveArray ) - 1 ]
825:          ENDIF
826:
827:          * Reset the error trap and return to home
828:          ON ERROR &olderror
829:          SET DEFAULT TO (m.olddefault)
830:
831:          * Initialize the popup variable to the element in the
832:          * drive array which contains the current drive
833:          mdrive = ASCAN( drivearray, m.olddefault )
834: <--------RETURN
835:
836:          **********
837:       ENDCASE
838:
839:       *
840:       *
841:       *          BACKUP Procedure NEWPATHPOP
842:       *
843:       *
844:       *
845:
846:
847:       PROCEDURE newpathpop
848:       DO CASE
849:       CASE  DOS
850:          **********
851:          * Create a popup with one bar for each subdirectory
852:          * in the current path
853:          PRIVATE dirstring, dircount
854:
855:          * Base the popup on an array with one element for
856:          * each subdirectory in the current path
857:          *
858:          * Start the array with the current drive
859:          DECLARE patharray[ 1 ]
860:          patharray[ 1 ] = SET( "DEFAULT" )
861:
862:          * Get the current path string
863:          m.dirstring = CURDIR()
864:
865:          * If we are not in the root directory
866:          IF LEN( m.dirstring ) > 1
867:
868:             * Start parsing the path string for each directory name
```

```
869:          m.dircount = 1
870:
871:          * Continue as long as there is a pair of slashes
872:          * Surrounding the next part of the string
873:          DO WHILE AT( "\", m.dirstring, m.dircount ) <> 0 AND ;
874:                AT( "\", m.dirstring, m.dircount + 1 ) <> 0
875:
876:             * Add another element at the end of the array
877:             DECLARE patharray[ m.DirCount + 1 ]
878:
879:             * Cut out the next set of characters between the slashes
880:             m.firstchar = AT( "\", m.dirstring, m.dircount ) + 1
881:
882:             m.pathlength = AT( "\", m.dirstring, m.dircount + 1 )  ;
883:                - m.firstchar
884:
885:             * And add them to the next array element
886:             patharray[ m.DirCount + 1 ] = SUBSTR( m.dirstring, ;
887:                m.firstchar, m.pathlength )
888:
889:             * Look for the next directory name in the path string
890:             m.dircount = m.dircount + 1
891:
892:          ENDDO
893:       ENDIF
894:
895:       * Point the popup variable at the last element in the array
896:       mpath = ALEN( patharray )
897: <-----RETURN
898:
899:       ********************
900:    CASE  WINDOWS
901:       ********************
902:       * Create a popup with one bar for each subdirectory
903:       * in the current path
904:       PRIVATE dirstring, dircount
905:
906:       * Base the popup on an array with one element for
907:       * each subdirectory in the current path
908:       *
909:       * Start the array with the current drive
910:       DECLARE patharray[ 1 ]
911:       patharray[ 1 ] = SET( "DEFAULT" )
912:
913:       * Get the current path string
914:       m.dirstring = CURDIR()
915:
916:       * If we are not in the root directory
917:       IF LEN( m.dirstring ) > 1
918:
919:          * Start parsing the path string for each directory name
920:          m.dircount = 1
921:
922:          * Continue as long as there is a pair of slashes
923:          * Surrounding the next part of the string
924:          DO WHILE AT( "\", m.dirstring, m.dircount ) <> 0 AND ;
925:                AT( "\", m.dirstring, m.dircount + 1 ) <> 0
926:
927:             * Add another element at the end of the array
928:             DECLARE patharray[ m.DirCount + 1 ]
929:
930:             * Cut out the next set of characters between the slashes
931:             m.firstchar = AT( "\", m.dirstring, m.dircount ) + 1
932:
933:             m.pathlength = AT( "\", m.dirstring, m.dircount + 1 )  ;
934:                - m.firstchar
935:
936:             * And add them to the next array element
937:             patharray[ m.DirCount + 1 ] = SUBSTR( m.dirstring, ;
938:                m.firstchar, m.pathlength )
939:
940:             * Look for the next directory name in the path string
941:             m.dircount = m.dircount + 1
942:
943:          ENDDO
944:       ENDIF
945:
946:       * Point the popup variable at the last element in the array
947:       mpath = ALEN( patharray )
948: <-----RETURN
949:
950:       ********************
951:    ENDCASE
952:
953:    *
954:    *
955:    *                BACKUP Procedure NEWFILEPOP
956:    *
957:    *
958:    *
959:
960:
961:    PROCEDURE newfilepop
962:    DO CASE
963:    CASE  DOS
964:       ********************
965:
966:       * Create a popup with all of the file names and its subdirection i
=> n the current directory
967:       * This will be based on an array as well
968:
969:       * Create an array with all the files matching the current wild car
=> d
970:       * ADIR() creates an array with 5 columns.
971:
972:       PRIVATE startsort
973:       * Fill an array with directory names only
974:       =ADIR( dirarray, "", "D")
975:       SIZE = ALEN(dirarray,1)
976:       IF dirarray[ 1, 1 ] = "."
977:          =ADEL(dirarray,1)
978:          SIZE = SIZE - 1
979:          DECLARE dirarray[size,5]
980:       ELSE
981:          * We must be in the root directory "\"
982:          * Add one more row to the directory array
983:          SIZE = SIZE + 1
984:          DECLARE dirarray[ SIZE, 5 ]
985:
986:          * Push all the rows down by one
987:          =AINS( dirarray, 1 )
988:
989:          * Fill in the first directory name in the array
990:          * a bug in the popups makes it refuse to display a
991:          * prompt of "\", use "\\" to make one \ appear
992:          * even then the bar will automatically be non-selectable
993:          * which in this case is fine
994:          dirarray[ 1,1 ] = "\\"
995:
996:          * Start the sort after the root name
997:       ENDIF
998:
```

```
 999:     IF ALEN( dirarray, 1 ) > 2
1000:
1001:        * Sort the array starting at the starting row
1002:        =ASORT( dirarray, AELEMENT( dirarray, 2, 1 ) )
1003:     ENDIF
1004:     FOR i = 1 TO ALEN(dirarray,1)
1005:        dirarray[i,1] = "[" + dirarray[i,1] + "]"
1006:     ENDFOR
1007:
1008:     IF ADIR(filearray, m.wildcard) = 0
1009:        SIZE = ALEN(dirarray,1)
1010:        DECLARE filearray[size,5]
1011:        =ACOPY(dirarray, filearray)
1012:     ELSE
1013:        stop = ALEN(dirarray,1)
1014:        FOR i = 1 TO stop
1015:           SIZE = ALEN(filearray,1)
1016:           DECLARE filearray[size + 1,5]
1017:           =AINS(filearray,1)
1018:           filearray[1,1] = dirarray(alen[dirarray,1],1)
1019:           IF ALEN(dirarray,1) > 1
1020:              DECLARE dirarray[alen(dirarray,1)-1,5]
1021:           ENDIF
1022:        ENDFOR
1023:     ENDIF
1024:     mfile = 1
1025: <===RETURN
1026:
1027:     ********************
1028:  CASE _WINDOWS
1029:     ********************
1030:
1031:     * Create a popup with all of the file names and its subdirection i
 => n the current directory
1032:     * This will be based on an array as well
1033:
1034:     * Create an array with all the files matching the current wild car
 => d
1035:     * ADIR() creates an array with 5 columns.
1036:
1037:     PRIVATE startsort
1038:     * Fill an array with directory names only
1039:     =ADIR( dirarray, "", "D")
1040:     SIZE = ALEN(dirarray,1)
1041:     IF dirarray[ 1, 1 ] = "."
1042:        =ADEL(dirarray,1)
1043:        SIZE = SIZE - 1
1044:        DECLARE dirarray[size,5]
1045:     ELSE
1046:        * We must be in the root directory "\"
1047:        * Add one more row to the directory array
1048:        SIZE = SIZE + 1
1049:        DECLARE dirarray[ SIZE, 5 ]
1050:
1051:        * Push all the rows down by one
1052:        =AINS( dirarray, 1 )
1053:
1054:        * Fill in the first directory name in the array
1055:        * a bug in the popups makes it refuse to display a
1056:        * prompt of "\", use "\\" to make one \ appear
1057:        * even then the bar will automatically be non-selectable
1058:        * which in this case is fine
1059:        dirarray[ 1,1 ] = "\\"
1060:
1061:        * Start the sort after the root name
1062:     ENDIF
```

```
1063:
1064:     IF ALEN( dirarray, 1 ) > 2
1065:
1066:        * Sort the array starting at the starting row
1067:        =ASORT( dirarray, AELEMENT( dirarray, 2, 1 ) )
1068:     ENDIF
1069:     FOR i = 1 TO ALEN(dirarray,1)
1070:        dirarray[i,1] = "[" + dirarray[i,1] + "]"
1071:     ENDFOR
1072:
1073:     IF ADIR(filearray, m.wildcard) = 0
1074:        SIZE = ALEN(dirarray,1)
1075:        DECLARE filearray[size,5]
1076:        =ACOPY(dirarray, filearray)
1077:     ELSE
1078:        stop = ALEN(dirarray,1)
1079:        FOR i = 1 TO stop
1080:           SIZE = ALEN(filearray,1)
1081:           DECLARE filearray[size + 1,5]
1082:           =AINS(filearray,1)
1083:           filearray[1,1] = dirarray(alen[dirarray,1],1)
1084:           IF ALEN(dirarray,1) > 1
1085:              DECLARE dirarray[alen(dirarray,1)-1,5]
1086:           ENDIF
1087:        ENDFOR
1088:     ENDIF
1089:     mfile = 1
1090: <===RETURN
1091:
1092:     ********************
1093:  ENDCASE
1094:
1095:  *
1096:  *
1097:  *                 BACKUP Function PATHSTRING
1098:  *
1099:  *
1100:  *
1101:
1102:
1103:  PROCEDURE pathstring
1104: DO CASE
1105: CASE _DOS
1106:     ********************
1107:     * Convert an array of subdirectories, such as PathArray,
1108:     * into a legal path string for use with SET DEFAULT
1109:     * Use the current value of the path popup as the
1110:     * ending point on the path
1111:     PRIVATE ppath
1112:
1113:     * Start with the drive name
1114:     m.ppath = patharray[ 1 ]
1115:
1116:     * If the path popup is pointing to a subdirectory
1117:     IF mpath > 1
1118:
1119:        * Add all the subdirectories to the path string
1120:        FOR i = 2 TO mpath
1121:           m.ppath = m.ppath + "\" + patharray[ i ]
1122:
1123:        NEXT
1124:     ENDIF
1125:
1126:     * End the path with one last slash for good luck
1127:     m.ppath = m.ppath + "\"
1128:
```

```
<-- RETURN m.ppath

    *******************
 CASE _WINDOWS
    *******************
    * Convert an array of subdirectories, such as PathArray,
    * into a legal path string for use with SET DEFAULT
    * Use the current value of the path popup as the
    * ending point on the path
    PRIVATE ppath

    * Start with the drive name
    m.ppath = patharray[ 1 ]

    * If the path popup is pointing to a subdirectory
   IF mpath > 1

      * Add all the subdirectories to the path string
     FOR i = 2 TO mpath
        m.ppath = m.ppath + "\" + patharray[ i ]

     NEXT
   ENDIF

    * End the path with one last slash for good luck
    m.ppath = m.ppath + "\"

<-- RETURN m.ppath

    *******************
 ENDCASE

  *
  *
  *        BACKUP Procedure PKZIP
  *
  *
  *


  PROCEDURE pkzip
 DO CASE
 CASE _DOS
     *******************

     PRIVATE CURDIR, datadir, msel
     CURDIR = FULLPATH(CURDIR())
    IF !EMPTY(GETENV("CAMD"))
        datadir = FULLPATH(GETENV("CAMD"))
    ELSE
        datadir = CURDIR()
    ENDIF
     desdir = pathstring() + mfname
     SET DEFA TO (datadir)
    IF !EMPTY(GETENV("CAMDUTIL"))
        pkzipcom = "!" + GETENV("CAMDUTIL") + "\PKZIP &desdir *.dbf *.c
=> dx *.idx *.fpt"
    ELSE
        pkzipcom = "!PKZIP &desdir *.dbf *.cdx *.idx *.fpt"
    ENDIF
     &pkzipcom
     USE
     DELETE FILE t0000000.txt
     SET DEFA TO (CURDIR)
<-- RETURN
 CASE _WINDOWS
     *******************

     PRIVATE CURDIR, datadir, msel
     CURDIR = FULLPATH(CURDIR())
    IF !EMPTY(GETENV("KBDATA"))
        datadir = FULLPATH(GETENV("KBDATA"))
    ELSE
        datadir = CURDIR()
    ENDIF
     desdir = pathstring() + mfname
     SET DEFA TO (datadir)
    IF !EMPTY(GETENV("CAMDUTIL"))
        pkzipcom = "!" + GETENV("CAMDUTIL") + "\PKZIP &desdir *.dbf *.c
=> dx *.idx *.fpt"
    ELSE
        pkzipcom = "!PKZIP &desdir *.dbf *.cdx *.idx *.fpt"
    ENDIF
     &pkzipcom
     USE
     DELETE FILE t0000000.txt
     SET DEFA TO (CURDIR)
<-- RETURN
 ENDCASE
*:  EOF: BACKUP.ac1
```

```
*
*       08/09/94                KB.SPR                  09:38:51
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              Windows Window definitions          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_kb") ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.PJX" ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.SCX" ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.MNX" ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.PRG" ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.FRX" ;
         OR UPPER(WTITLE("W_KB")) == "W_KB.QPR"
      DEFINE WINDOW w_kb ;
         AT  0.000, 0.000  ;
         SIZE 30.167,69.250 ;
         TITLE "Knowledge Base Area" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE
      MOVE WINDOW w_kb CENTER
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │           KB/Windows Setup Code - SECTION 2      │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   PRIVATE areaname, areaid
   PRIVATE mquals, mgoals, mdispl
   DIMENSION mquals[10,4],mgoals[10,4],mdispl[10,4]
   DIMENSION methods[4]
   DIMENSION infs[2]
   methods[1] = "BAYES"
   methods[2] = "ADDITIVE"
   methods[3] = "FUZZY"
   methods[4] = "NONE"
   infs[1]    = "FORWARD"
   infs[2]    = "BACKWARD"

   SELECT 0
   USE area INDEX area
   SELECT 0
   USE disease INDEX disid, disease
   SELECT 0
   USE enum INDEX enum
   SELECT 0
   USE VAL INDEX VAL
   SELECT 0
   USE dict INDEX dictid, dictname
   SET RELATION TO id INTO enum, id INTO VAL
   SELECT 0
   USE goals INDEX goals
   SET FILTER TO area = area.area
   SELECT 0
   USE DISPLAY
   SET FILTER TO area = area.area
   SELECT 0
   USE quals INDEX quals
   SET FILTER TO area = area.area
   SELECT area
   SCATTER MEMVAR
   m.areaname = m.name
   m.areaid = m.area
   DO setquals
   DO setgoals
   DO setdisplay
   SELECT area
   m.mpop = area.method + 1
   m.mpop2 = area.inference + 1


   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              KB/Windows Screen Layout            │
   *       │
   *
```

```
   *        ┌──────────────────────────────
   *
   #REGION 1
  ┌IF WVISIBLE("w_kb")
  │    ACTIVATE WINDOW w_kb SAME
  ├ELSE
  │    ACTIVATE WINDOW w_kb NOSHOW
  └ENDIF
   @ 9.250,4.500 SAY "Consider"  ;
      FONT "Terminal", 8
   @ 14.000,4.750 SAY "Probable"  ;
      FONT "Terminal", 8
   @ 18.417,5.000 SAY "Likely"  ;
      FONT "Terminal", 8
   @ 7.333,3.250 TO 21.666,24.875 ;
      PEN 1, 8
   @ 5.833,4.000 SAY "Display thresholds"  ;
      FONT "Terminal", 8
   @ 15.917,26.250 SAY "Inference"  ;
      FONT "Terminal", 8
   @ 20.583,26.000 SAY "Confidence"  ;
      FONT "Terminal", 8
   @ 1.500,4.125 SAY "Knowledge Base Area:"  ;
      FONT "Terminal", 8 ;
      STYLE "T"
   @ 8.500,26.375 SAY "Total Rules:"  ;
      FONT "Terminal", 8 ;
      STYLE "T"
   @ 25.083,5.625 GET mbuttons ;
      PICTURE "@*HN \<Add;\<Edit;\<Delete;\<OK;\<Cancel" ;
      SIZE 1.917,11.125,0.875 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kofx9()
   @ 6.083,53.500 GET mbbuttons ;
      PICTURE "@*VN \<Next;\<Previous;\<Browse;\<Qualifiers;\<Goals;C
=> onc\<Lusion" ;
      SIZE 2.000,11.625,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kog2m()
   @ 1.583,24.875 GET m.areaname ;
      SIZE 1.000,34.750 ;
      DEFAULT " " ;
      FONT "Terminal", 8
   @ 9.250,13.625 GET m.threshold ;
      SIZE 1.333,8.750 ;
      DEFAULT 0 ;
      FONT "Terminal", 8
   @ 14.000,13.875 GET m.probable ;
      SIZE 1.333,9.000 ;
      DEFAULT 0 ;
      FONT "Terminal", 8
   @ 18.500,13.875 GET m.likely ;
      SIZE 1.417,9.250 ;
      DEFAULT 0 ;
      FONT "Terminal", 8
   @ 8.500,39.750 GET m.rules ;
      SIZE 1.000,5.875 ;
      DEFAULT 0 ;
      FONT "Terminal", 8 ;
      DISABLE
   @ 11.917,26.500 GET m.isdiag ;
      PICTURE "@*C DIAGNOSIS" ;
      SIZE 1.417,11.875 ;
      DEFAULT 0 ;
      FONT "Terminal", 8
   @ 20.250,37.500 GET m.mpop ;
      PICTURE "@^" ;
      FROM methods ;
      SIZE 1.500,13.000 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kogfv()
   @ 15.833,37.500 GET m.mpop2 ;
      PICTURE "@^" ;
      FROM infs ;
      SIZE 1.500,12.750 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kogih()

  ┌IF NOT WVISIBLE("w_kb")
  │   ACTIVATE WINDOW w_kb
  └ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_kb

   #REGION 0

   SET readborder &rborder

  ┌IF m.talkstat = "ON"
  │   SET TALK ON
  └ENDIF
  ┌IF m.compstat = "ON"
  │   SET COMPATIBLE ON
  └ENDIF

   *        ┌──────────────────────────────
   *        │
   *        │      KB/Windows Cleanup Code
   *        │
   *        └──────────────────────────────
   *

   #REGION 1
   SELECT area
   USE
   SELECT goals
   USE
   SELECT DISPLAY
   USE
   SELECT quals
   USE
   SELECT disease
   USE
   SELECT enum
   USE
   SELECT dict
   USE
```

```
    SELECT VAL
    USE
<---RETURN

CASE _DOS

    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
       SET TALK OFF
       m.talkstat = "ON"
    ELSE
       m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    *
    *
    *          MS-DOS Window definitions
    *
    *
    *

    IF NOT WEXIST("w_kb") ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.PJX" ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.SCX" ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.MNX" ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.PRG" ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.FRX" ;
          OR UPPER(WTITLE("W_KB")) == "W_KB.QPR"
       DEFINE WINDOW w_kb ;
          FROM INT((SROW()-18)/2),INT((SCOL()-75)/2) ;
          TO INT((SROW()-18)/2)+17,INT((SCOL()-75)/2)+74 ;
          TITLE "Knowledge Base Area" ;
          FLOAT ;
          CLOSE ;
          SHADOW ;
          NOMINIMIZE ;
          COLOR SCHEME 1
    ENDIF

    *
    *
    *          KB/MS-DOS Setup Code - SECTION 2
    *
    *
    *

    #REGION 1
    SCATTER MEMVAR
    m.mpop = m.method + 1
    m.mpop2 = m.inference + 1

    *
    *
    *          KB/MS-DOS Screen Layout
    *
    *
    *

    #REGION 1
    IF WVISIBLE("w_kb")
       ACTIVATE WINDOW w_kb SAME
    ELSE
       ACTIVATE WINDOW w_kb NOSHOW
    ENDIF
    @ 5,2 SAY "Consider" ;
       SIZE 1,8, 0
    @ 7,2 SAY "Probable" ;
       SIZE 1,8, 0
    @ 9,2 SAY "Likely" ;
       SIZE 1,6, 0
    @ 14,6 GET mbuttons ;
       PICTURE "@*HT \<Add;\<Edit;\<Delete;\<OK;\<Cancel" ;
       SIZE 1,12,1 ;
       DEFAULT 1 ;
       VALID _qsf0koh4z()
    @ 5,12 GET m.threshold ;
       SIZE 1,6 ;
       DEFAULT 0
    @ 7,12 GET m.probable ;
       SIZE 1,6 ;
       DEFAULT 0
    @ 9,12 GET m.likely ;
       SIZE 1,6 ;
       DEFAULT 0
    @ 1,17 GET m.name ;
       SIZE 1,34 ;
       DEFAULT " "
    @ 3,0 TO 10,24
    @ 3,4 SAY "Display thresholds" ;
       SIZE 1,18, 0
    @ 3,28 SAY "Rules" ;
       SIZE 1,5, 0
    @ 3,37 GET m.rules ;
       SIZE 1,4 ;
       DEFAULT " " ;
       DISABLE
    @ 9,39 GET m.mpop ;
       PICTURE "@^" ;
       FROM methods ;
       SIZE 3,12 ;
       DEFAULT 1 ;
       VALID _qsf0kohj1() ;
       COLOR SCHEME 1, 2
    @ 7,28 SAY "Inference" ;
       SIZE 1,9, 0
    @ 6,39 GET m.mpop2 ;
       PICTURE "@^" ;
       FROM infs ;
       SIZE 3,12 ;
```

```
361:          DEFAULT 1 ;
362:          VALID _qsf0kohm5() ;
363:          COLOR SCHEME 1, 2
364:       @ 10,28 SAY "Confidence" ;
365:          SIZE 1,10, 0
366:       @ 5,28 GET m.isdiag ;
367:          PICTURE "@*C DIAGNOSIS" ;
368:          SIZE 1,13 ;
369:          DEFAULT 0
370:       @ 1,0 GET minvbutton ;
371:          PICTURE "@*HN \<Knowledge Base" ;
372:          SIZE 1,16,1 ;
373:          DEFAULT 1 ;
374:          VALID _qsf0kohpt() ;
375:          MESSAGE "Knowledge Base"
376:       @ 2,58 GET mbbuttons ;
377:          PICTURE "@*VN \<Next;\<Previous;\<Browse;\<Qualifiers;\<Goals;C
=> onc\<Lusion" ;
378:          SIZE 1,13,1 ;
379:          DEFAULT 1
380:
381:       IF NOT WVISIBLE("w_kb")
382:          ACTIVATE WINDOW w_kb
383:       ENDIF
384:
385:       READ CYCLE MODAL
386:
387:       RELEASE WINDOW w_kb
388:
389:       #REGION 0
390:       IF m.talkstat = "ON"
391:          SET TALK ON
392:       ENDIF
393:       IF m.compstat = "ON"
394:          SET COMPATIBLE ON
395:       ENDIF
396:
397:
398:    ENDCASE
399:
400:
401:    *
402:    *
403:    *        _QSF0KOFX9            mbuttons VALID
404:    *
405:    *        Function Origin:
406:    *
407:    *        From Platform:        Windows
408:    *        From Screen:          KB,     Record Number:   11
409:    *        Variable:             mbuttons
410:    *        Called By:            VALID Clause
411:    *        Object Type:          Push Button
412:    *        Snippet Number:       1
413:    *
414:    *
415:    *
416:    FUNCTION _qsf0kofx9     &&  mbuttons VALID
417:    #REGION 1
418:    DO CASE
419:    CASE mbuttons = 1             && <Add>
420:       SET TOPIC TO "ADD"
421:       DO kbload.spr
422:    CASE mbuttons = 2                      && <Edit>
423:       SET TOPIC TO "kbedit"
424:       DO kbedit.spr
425:    CASE mbuttons = 3             && <Delete>
426:       SET TOPIC TO "DELETE"
427:       DO kbdel.spr
428:    CASE mbuttons = 4             && <ok>
429:       SELECT area
430:       m.name = m.areaname
431:       GATHER MEMVAR
432:       CLEAR READ
433:    CASE mbuttons = 5             && <cancel>
434:       CLEAR READ
435:    ENDCASE
436:    SELECT area
437:    SCATTER MEMVAR
438:    DO setquals
439:    DO setgoals
440:    DO setdisplay
441:    m.mpop = area.method + 1
442:    m.mpop2 = area.inference + 1
443:    SHOW GETS
444:    mtopic = ALIAS()
445:    SET TOPIC TO &mtopic
446:
447:
448:    *
449:    *
450:    *        _QSF0KOG2M            mbuttons VALID
451:    *
452:    *        Function Origin:
453:    *
454:    *        From Platform:        Windows
455:    *        From Screen:          KB,     Record Number:   12
456:    *        Variable:             mbuttons
457:    *        Called By:            VALID Clause
458:    *        Object Type:          Push Button
459:    *        Snippet Number:       2
460:    *
461:    *
462:    *
463:    FUNCTION _qsf0kog2m     &&  mbuttons VALID
464:    #REGION 1
465:    SELECT area
466:    DO CASE
467:    CASE mbuttons = 1             && <Next>
468:       m.recno = RECNO()
469:       IF !EOF()
470:          SKIP
471:          SHOW GET mbbuttons,2 ENABLE
472:          IF EOF()
473:             GOTO BOTTOM
474:             SHOW GET mbbuttons,1 DISABLE
475:          ENDIF
476:       ELSE
477:          GOTO BOTTOM
478:          SHOW GET mbbuttons,2 ENABLE
479:          SHOW GET mbbuttons,1 DISABLE
480:       ENDIF
481:    CASE mbuttons = 2             && <previous>
482:       m.recno = RECNO()
483:       IF !BOF()
484:          SKIP -1
485:          SHOW GET mbbuttons,1 ENABLE
486:       ELSE
487:          GOTO TOP
488:          SHOW GET mbbuttons,1 ENABLE
489:          SHOW GET mbbuttons,2 DISABLE
490:       ENDIF
491:    CASE mbuttons = 3             && <Browse>
```

```
492:        SET TOPIC TO "BROWSE"
493:        BROWSE NOEDIT
494:     CASE mbbuttons = 4              && <Qualifiers>
495:        PRIVATE msel
496:        msel = SELECT()
497:        SELECT quals
498:        SET TOPIC TO "QUALIFIERS"
499:        DO qual.spr
500:        SELECT (msel)
501: <------RETURN .T.
502:     CASE mbbuttons = 5              && <Goals>
503:        PRIVATE msel
504:        msel = SELECT()
505:        SELECT goals
506:        SET TOPIC TO "GOALS"
507:        DO goal.spr
508:        SELECT (msel)
509: <------RETURN .T.
510:     CASE mbbuttons = 6              && <conclusion>
511:        PRIVATE msel
512:        msel = SELECT()
513:        SELECT DISPLAY
514:        SET TOPIC TO "DISPLAY"
515:        DO display.spr
516:        SELEC (msel)
517: <------RETURN .T.
518:     ENDCASE
519:     SCATTER MEMVAR
520:     m.areaname = m.name
521:     m.areaid = m.area
522:     DO setquals
523:     DO setgoals
524:     DO setdisplay
525:     m.mpop = area.method + 1
526:     m.mpop2 = area.inference + 1
527:     SHOW GETS
528:     mtopic = ALIAS()
529:     SET TOPIC TO &mtopic
530:
531:
532:     *
533:     *
534:     *          _QSFOKOGFV            m.mpop VALID
535:     *
536:     *          Function Origin:
537:     *
538:     *          From Platform:        Windows
539:     *          From Screen:          KB,     Record Number:   19
540:     *          Variable:             m.mpop
541:     *          Called By:            VALID Clause
542:     *          Object Type:          Popup
543:     *          Snippet Number:       3
544:     *
545:     *
546:     *
547:     FUNCTION _qsf0kogfv     &&  m.mpop VALID
548:     #REGION 1
549:     m.method = m.mpop - 1
550:
551:
552:     *
553:     *
554:     *          _QSFOKOGIH            m.mpop2 VALID
555:     *
556:     *          Function Origin:
557:     *
```



```
558:     *          From Platform:        Windows
559:     *          From Screen:          KB,     Record Number:   20
560:     *          Variable:             m.mpop2
561:     *          Called By:            VALID Clause
562:     *          Object Type:          Popup
563:     *          Snippet Number:       4
564:     *
565:     *
566:     *
567:     FUNCTION _qsf0kogih     &&  m.mpop2 VALID
568:     #REGION 1
569:     m.inference = m.mpop2 - 1
570:
571:
572:     *
573:     *
574:     *          _QSFOKOH4Z            mbuttons VALID
575:     *
576:     *          Function Origin:
577:     *
578:     *          From Platform:        MS-DOS
579:     *          From Screen:          KB,     Record Number:   26
580:     *          Variable:             mbuttons
581:     *          Called By:            VALID Clause
582:     *          Object Type:          Push Button
583:     *          Snippet Number:       5
584:     *
585:     *
586:     *
587:     FUNCTION _qsf0koh4z     &&  mbuttons VALID
588:     #REGION 1
589:     DO CASE
590:     CASE mbuttons = 1               && <Add>
591:        SET TOPIC TO "ADD"
592:        DO kbload.spr
593:     CASE mbuttons = 2               && <Delete>
594:        SET TOPIC TO "DELETE"
595:        DO kbdel.spr
596:     CASE mbuttons = 3               && <conclusion>
597:        SELECT DISPLAY
598:        SET TOPIC TO "DISPLAY"
599:        DO display.spr
600: <------RETURN .T.
601:     CASE mbuttons = 4               && ok
602:        GATHER MEMVAR
603:     CASE mbuttons = 5               && cancel
604:        SCATTER MEMVAR
605:     ENDCASE
606:
607:     DO CASE
608:     CASE mbbutton = 1 .OR. mbbutton = 2      && <Next> <Previous>
609:        IF ALIAS() != "AREA" .AND. ALIAS() != "RULE"
610: <---------RETURN .T.
611:        ENDIF
612:        m.recno = RECNO()
613:        IF mbbutton = 1  .AND. !EOF()         && <Next>
614:           SKIP
615:        ENDIF
616:        IF mbbutton = 2  .AND.  !BOF()        && <Previous>
617:           SKIP -1
618:        ENDIF
619:        DO CASE
620:        CASE ALIAS() = "RULE"
621:           m.goback  = area != area.area
622:        OTHERWISE
623:           m.goback = EOF()
```

```
624:     ENDCASE
625:     IF m.goback
626:        GOTO m.recno
627:     ENDIF
628:  CASE mbbutton = 3                          && <Browse>
629:     SET TOPIC TO "BROWSE"
630:     DO CASE
631:     CASE ALIAS() = "AREA"
632:        BROWSE NOEDIT
633:     CASE ALIAS() = "RULE"
634:        BROWSE FOR area = area.area NOEDIT
635:     ENDCASE
636:  CASE mbbutton = 4                          && <Edit>
637:     SET TOPIC TO "EDIT"
638:     DO CASE
639:     CASE ALIAS() = "AREA"
640:        DO kb.spr
641:     CASE ALIAS() = "RULE"
642:        DO rule.spr
643:     ENDCASE
644:  CASE mbbutton = 5
645:     SELECT quals
646:     SET TOPIC TO "QUALIFIERS"
647:     DO qual.spr
648: <-------RETURN .T.
649:  CASE mbbutton = 6
650:     SELECT goals
651:     SET TOPIC TO "GOALS"
652:     DO goal.spr
653: <-------RETURN .T.
654:  ENDCASE
655:  DO setprem
656:  DO setact
657:  DO setquals
658:  DO setgoals
659:  DO setdisplay
660:  m.mpop = area.method + 1
661:  m.mpop2 = area.inference + 1
662:  SHOW GETS
663:  mtopic = ALIAS()
664:  SET TOPIC TO &mtopic
665:
666:
667:  *
668:  *
669:  *        _QSF0KOHJ1            m.mpop VALID
670:  *
671:  *        Function Origin:
672:  *
673:  *        From Platform:        MS-DOS
674:  *        From Screen:          KB,     Record Number:   35
675:  *        Variable:             m.mpop
676:  *        Called By:            VALID Clause
677:  *        Object Type:          Popup
678:  *        Snippet Number:       6
679:  *
680:  *
681:  *
682:  FUNCTION _qsf0kohj1     &&  m.mpop VALID
683:  #REGION 1
684:  m.method = m.mpop - 1
685:
686:
687:  *
688:  *
689:  *        _QSF0KOHM5            m.mpop2 VALID
```

```
690:  *
691:  *        Function Origin:
692:  *
693:  *        From Platform:        MS-DOS
694:  *        From Screen:          KB,     Record Number:   37
695:  *        Variable:             m.mpop2
696:  *        Called By:            VALID Clause
697:  *        Object Type:          Popup
698:  *        Snippet Number:       7
699:  *
700:  *
701:  *
702:  FUNCTION _qsf0kohm5     &&  m.mpop2 VALID
703:  #REGION 1
704:  m.inference = m.mpop2 - 1
705:
706:
707:  *
708:  *
709:  *        _QSF0KOHPT            minvbutton VALID
710:  *
711:  *        Function Origin:
712:  *
713:  *        From Platform:        MS-DOS
714:  *        From Screen:          KB,     Record Number:   40
715:  *        Variable:             minvbutton
716:  *        Called By:            VALID Clause
717:  *        Object Type:          Push Button
718:  *        Snippet Number:       8
719:  *
720:  *
721:  *
722:  FUNCTION _qsf0kohpt     &&  minvbutton VALID
723:  #REGION 1
724:  SELECT area
725:  SET TOPIC TO "AREA"
726:
727:
728:
729:  *
730:  *
731:  *        KB/MS-DOS Supporting Procedures and Functions
732:  *
733:  *
734:  *
735:
736:  #REGION 1
737:
738:  *
739:  *
740:  *                    KB Procedure SETQUALS
741:  *
742:  *
743:  *
744:
745:  PROCEDURE setquals
746:  PRIVATE msel,i
747:  msel = SELECT()
748:  SELECT quals
749:  nq = RECCOUNT()
750:  nq = MAX(nq,1)
751:  DIMENSION mquals[nq,4]
752:  GOTO TOP
753:  mquals = ""
754:   i =0
755: DO WHILE !EOF()
```

```
        i = i + 1
        mquals[i,3]  = OBJECT
        mquals[i,4]  = id
        IF OBJECT = "D"
            SELECT disease
        ELSE
            SELECT dict
        ENDIF
        SEEK mquals[i,4]
        mquals[i,1] = name
        mquals[i,2] = ALIAS
        SELECT quals
        SKIP
    ENDDO
    nq = i
    SELECT (msel)
<---RETURN


    *
    *                    KB Procedure SETGOALS
    *
    *
    *

    PROCEDURE setgoals
    PRIVATE msel, i
    msel = SELECT()
    SELECT goals
    ng = RECCOUNT()
    ng = MAX(ng,1)
    DIMENSION mgoals[ng,4]
    GOTO TOP
    i = 0
    mgoals = ""
    DO WHILE !EOF()
        i = i + 1
        mgoals[i,3]  = OBJECT
        mgoals[i,4]  = id
        IF OBJECT = "D"
            SELECT disease
        ELSE
            SELECT dict
        ENDIF
        SEEK mgoals[i,4]
        mgoals[i,1] = name
        mgoals[i,2] = ALIAS
        SELECT goals
        SKIP
    ENDDO
    ng = i
    SELECT (msel)
<---RETURN


    *
    *                    KB Procedure SETDISPLAY
    *
    *
    *

    PROCEDURE setdisplay
    PRIVATE msel, i
    msel = SELECT()
    SELECT DISPLAY
    nd = RECCOUNT()
    nd = MAX(nd,1)
    DIMENSION mdispl[nd,4]
    GOTO TOP
    i = 0
    mdispl = ""
    DO WHILE !EOF()
        i = i + 1
        mdispl[i,3]  = OBJECT
        mdispl[i,4]  = id
        IF OBJECT = "D"
            SELECT disease
        ELSE
            SELECT dict
        ENDIF
        SEEK mdispl[i,4]
        mdispl[i,1] = name
        mdispl[i,2] = ALIAS
        SELECT DISPLAY
        SKIP
    ENDDO
    nd = i
    SELECT (msel)
<---RETURN


    *
    *                    KB Procedure EDGOAL
    *
    *
    *

    PROCEDURE edgoal
    =edobj( mgoals[mg,3], mgoals[mg,4] )
<---RETURN


    *
    *                    KB Procedure EDQUAL
    *
    *
    *

    PROCEDURE edqual
    =edobj( mquals[mq,3], mquals[mq,4] )
<---RETURN


    *
    *                    KB Procedure EDDISPLAY
    *
    *
    *

    PROCEDURE eddisplay
    =edobj( mdispl[md,3], mdispl[md,4] )
<---RETURN


    *
    *
    *                    KB Function EDOBJ
```

```
888:     *
889:     *
890:     *
891:
892:     FUNCTION edobj
893:     PARAMETERS mobj, mid
894:     PRIVATE msel
895:     msel = SELECT()
896:    ┌IF mobj = "D"
897:    │   SELECT disease
898:    │   SEEK mid
899:    │   DO disease.spr
900:    ├ELSE
901:    │   SELECT dict
902:    │   SEEK mid
903:    │   DO dict.spr WITH mid
904:    └ENDIF
905:     SELECT (msel)
906: <──RETURN ""
907:
908:
909:  *:  EOF: KB.ac1
```

```
*
*
*    08/09/94            KBDEL.SPR            09:38:58
*
*
*
*    Author's Name
*
*    Copyright (c) 1994 Company Name
*    Address
*    City,     Zip
*
*    Description:
*    This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *
   *
   *              Windows Window definitions
   *
   *
   *

   IF NOT WEXIST("_qsf0kok6q")
      DEFINE WINDOW _qsf0kok6q ;
         AT  0.000, 0.000  ;
         SIZE 7.917,42.500 ;
         TITLE "Knowledge Base Delete" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         NOCLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         DOUBLE
      MOVE WINDOW _qsf0kok6q CENTER
   ENDIF


   *
   *
   *              KBDEL/Windows Screen Layout
   *
   *
   *

   #REGION 1
   IF WVISIBLE("_qsf0kok6q")
      ACTIVATE WINDOW _qsf0kok6q SAME
   ELSE
      ACTIVATE WINDOW _qsf0kok6q NOSHOW
   ENDIF
   @ 4.250,10.375 GET mbutton ;
      PICTURE "@*HT OK;Cancel" ;
      SIZE 2.000,8.250,1.500 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kokeu()
   @ 1.333,9.125 GET area.name ;
      SIZE 1.000,31.250 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      DISABLE
   @ 1.250,0.750 SAY "Delete:"  ;
      FONT "Terminal", 8

   IF NOT WVISIBLE("_qsf0kok6q")
      ACTIVATE WINDOW _qsf0kok6q
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW _qsf0kok6q

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


CASE _DOS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   *
```

```
*
*               MS-DOS Window definitions
*
*
*

IF NOT WEXIST("_qsf0kokln")
   DEFINE WINDOW _qsf0kokln ;
      FROM INT((SROW()-8)/2),INT((SCOL()-39)/2) ;
      TO INT((SROW()-8)/2)+7,INT((SCOL()-39)/2)+38 ;
      TITLE "Knowledge Base Delete" ;
      FLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      DOUBLE ;
      COLOR SCHEME 5
ENDIF


*
*
*               KBDEL/MS-DOS Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("_qsf0kokln")
   ACTIVATE WINDOW _qsf0kokln SAME
ELSE
   ACTIVATE WINDOW _qsf0kokln NOSHOW
ENDIF
@ 4,8 GET mbutton ;
   PICTURE "@*HT OK;Cancel" ;
   SIZE 1,8,2 ;
   DEFAULT 1 ;
   VALID _qsf0kokpt()
@ 1,10 GET area.name ;
   SIZE 1,26 ;
   DEFAULT " " ;
   DISABLE
@ 1,1 SAY "Delete:" ;
   SIZE 1,7, 0

IF NOT WVISIBLE("_qsf0kokln")
   ACTIVATE WINDOW _qsf0kokln
ENDIF

READ CYCLE MODAL

RELEASE WINDOW _qsf0kokln

#REGION 0
IF m.talkstat = "ON"
   SET TALK ON
ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


ENDCASE


*
*
*    _QSFOKOKEU            mbutton VALID
*
*    Function Origin:
*
*    From Platform:       Windows
*    From Screen:         KBDEL,     Record Number:    2
*    Variable:            mbutton
*    Called By:           VALID Clause
*    Object Type:         Push Button
*    Snippet Number:      1
*
*
*
FUNCTION _qsf0kokeu     &&  mbutton VALID
#REGION 1
DO CASE
CASE mbutton = 1
   *do (m.home + "\kbdelete") with area.name
   DO kbdelete WITH area.name
ENDCASE
<==RETURN .T.

*
*
*    _QSFOKOKPT            mbutton VALID
*
*    Function Origin:
*
*    From Platform:       MS-DOS
*    From Screen:         KBDEL,     Record Number:    7
*    Variable:            mbutton
*    Called By:           VALID Clause
*    Object Type:         Push Button
*    Snippet Number:      2
*
*
*
FUNCTION _qsf0kokpt     &&  mbutton VALID
#REGION 1
DO CASE
CASE mbutton = 1
   *do (m.home + "\kbdelete") with area.name
   DO kbdelete WITH area.name
ENDCASE
<==RETURN .T.
*:  EOF: KBDEL.ac1
```

```
*
*       08/09/94              QUAL.SPR                  09:39:00
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │                Windows Window definitions        │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_qe") ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.PJX" ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.SCX" ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.MNX" ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.PRG" ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.FRX" ;
         OR UPPER(WTITLE("W_QE")) == "W_QE.QPR"
      DEFINE WINDOW w_qe ;
         AT  0.000, 0.000  ;
         SIZE 19.923,57.333 ;
         TITLE "Qualifier Editor" ;
         FONT "MS Sans Serif", 8 ;
         STYLE "B" ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,128,128,128)
      MOVE WINDOW w_qe CENTER
   ENDIF


   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │          QUAL/Windows Setup Code - SECTION 2     │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   EXTERNAL ARRAY mquals

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              QUAL/Windows Screen Layout          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_qe")
      ACTIVATE WINDOW w_qe SAME
   ELSE
      ACTIVATE WINDOW w_qe NOSHOW
   ENDIF
   @ 17.538,16.333 GET mbuttons ;
      PICTURE "@*HT \<Edit;\<Quit" ;
      SIZE 1.846,10.667,1.000 ;
      DEFAULT 1 ;
      FONT "MS Sans Serif", 8 ;
      STYLE "B" ;
      VALID _qsf0komjf()
   @ 0.385,2.667 GET mq ;
      PICTURE "@&N" ;
      FROM mquals ;
      SIZE 16.154,63.200 ;
      DEFAULT 1 ;
      FONT "MS Sans Serif", 8 ;
      VALID _qsf0komrx()

   IF NOT WVISIBLE("w_qe")
      ACTIVATE WINDOW w_qe
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_qe

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
```

```
    ENDIF
  IF m.compstat = "ON"
     SET COMPATIBLE ON
  ENDIF

  *
  *
  *        QUAL/Windows Cleanup Code
  *
  *
  *

  #REGION 1
  RETURN

CASE _DOS

  #REGION 0
  REGIONAL m.currarea, m.talkstat, m.compstat

  IF SET("TALK") = "ON"
     SET TALK OFF
     m.talkstat = "ON"
  ELSE
     m.talkstat = "OFF"
  ENDIF
  m.compstat = SET("COMPATIBLE")
  SET COMPATIBLE FOXPLUS

  *
  *
  *        MS-DOS Window definitions
  *
  *
  *

  IF NOT WEXIST("w_qe") ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.PJX" ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.SCX" ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.MNX" ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.PRG" ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.FRX" ;
        OR UPPER(WTITLE("W_QE")) == "W_QE.QPR"
     DEFINE WINDOW w_qe ;
        FROM INT((SROW()-17)/2),INT((SCOL()-58)/2) ;
        TO INT((SROW()-17)/2)+16,INT((SCOL()-58)/2)+57 ;
        TITLE "Qualifier Editor" ;
        FLOAT ;
        CLOSE ;
        SHADOW ;
        NOMINIMIZE ;
        COLOR SCHEME 1
  ENDIF


  *
  *
  *        QUAL/MS-DOS Setup Code - SECTION 2
  *
  *
  *

  #REGION 1
  EXTERNAL ARRAY mquals

  *
  *
  *        QUAL/MS-DOS Screen Layout
  *
  *
  *

  #REGION 1
  IF WVISIBLE("w_qe")
     ACTIVATE WINDOW w_qe SAME
  ELSE
     ACTIVATE WINDOW w_qe NOSHOW
  ENDIF
  @ 14,11 GET mbuttons ;
     PICTURE "@*HN \<Add;\<Delete;\<OK;\<Cancel" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0kon7e()
  @ 0,1 GET mq ;
     PICTURE "@&N" ;
     FROM mquals ;
     SIZE 13,54 ;
     DEFAULT 1 ;
     VALID _qsf0konbu() ;
     COLOR SCHEME 2

  IF NOT WVISIBLE("w_qe")
     ACTIVATE WINDOW w_qe
  ENDIF

  READ CYCLE MODAL

  RELEASE WINDOW w_qe

  #REGION 0
  IF m.talkstat = "ON"
     SET TALK ON
  ENDIF
  IF m.compstat = "ON"
     SET COMPATIBLE ON
  ENDIF


  *
```

```
   *
   *      QUAL/MS-DOS Cleanup Code
   *
   *
   *

   #REGION 1
   RETURN
   *------------------------------------------------------
   *
   * display popup notice
   *
   *------------------------------------------------------

ENDCASE


   *
   *
   *        _QSFOKOMJF           mbuttons VALID
   *
   *        Function Origin:
   *
   *        From Platform:       Windows
   *        From Screen:         QUAL,     Record Number:    2
   *        Variable:            mbuttons
   *        Called By:           VALID Clause
   *        Object Type:         Push Button
   *        Snippet Number:      1
   *
   *
   *
FUNCTION _qsf0komjf      &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && edit
   IF !EMPTY(mq)
      DO edqual
   ENDIF
   mok = .T.
CASE mbuttons = 2   && quit
   mok = .F.
   CLEAR READ
ENDCASE
SHOW GETS

   *
   *
   *        _QSFOKOMRX           mq VALID
   *
   *        Function Origin:
   *
   *        From Platform:       Windows
   *        From Screen:         QUAL,     Record Number:    3
   *        Variable:            mq
   *        Called By:           VALID Clause
   *        Object Type:         List
   *        Snippet Number:      2
   *
   *
   *
FUNCTION _qsf0komrx      &&  mq VALID
#REGION 1
DO edqual

   *
   *
   *        _QSFOKON7E           mbuttons VALID
   *
   *        Function Origin:
   *
   *        From Platform:       MS-DOS
   *        From Screen:         QUAL,     Record Number:    7
   *        Variable:            mbuttons
   *        Called By:           VALID Clause
   *        Object Type:         Push Button
   *        Snippet Number:      3
   *
   *
   *
FUNCTION _qsf0kon7e      &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && add
   DEACTIVATE WINDOW w_qe
   DO object.spr
   ACTIVATE WINDOW w_qe
   DO setquals
CASE mbuttons = 2   && delete
   mdel = validobj()
   IF mdel
      = qualdel()
   ENDIF
CASE mbuttons = 3   && ok
   mok = .T.
   CLEAR READ
CASE mbuttons = 4   && cancel
   mok = .F.
   CLEAR READ
ENDCASE
SHOW GETS

   *
   *
   *        _QSFOKONBU           mq VALID
   *
   *        Function Origin:
   *
   *        From Platform:       MS-DOS
   *        From Screen:         QUAL,     Record Number:    8
   *        Variable:            mq
   *        Called By:           VALID Clause
   *        Object Type:         List
   *        Snippet Number:      4
   *
   *
   *
FUNCTION _qsf0konbu      &&  mq VALID
#REGION 1
DO edqual


   *
   *
   *    QUAL/MS-DOS Supporting Procedures and Functions
   *
   *
   *
```

```
357:
358:     #REGION 1
359:
360:     *
361:     *
362:     *                    QUAL Function POPUPSHOW
363:     *
364:     *
365:     *
366:
367:     FUNCTION popupshow
368:     PARAMETERS errstr
369:    ┌IF NOT WEXIST("w_popnote")
370:    │   DEFINE WINDOW w_popnote ;
371:    │       FROM INT((SROW()-8)/2),INT((SCOL()-36)/2) ;
372:    │       TO INT((SROW()-8)/2)+7,INT((SCOL()-36)/2)+35 ;
373:    │       TITLE "One moment" ;
374:    │       FLOAT ;
375:    │       CLOSE ;
376:    │       SHADOW ;
377:    │       DOUBLE ;
378:    │       COLOR SCHEME 1
379:    └ENDIF
380:    ┌IF WVISIBLE("w_popnote")
381:    │   ACTIVATE WINDOW w_popnote SAME
382:    ├ELSE
383:    │   ACTIVATE WINDOW w_popnote NOSHOW
384:    └ENDIF
385:     @ 1,1 SAY errstr SIZE 3,31
386:
387:    ┌IF NOT WVISIBLE("w_popnote")
388:    │   ACTIVATE WINDOW w_popnote
389:    └ENDIF
390: <═══RETURN ""
391:
392:
393:     *
394:     *
395:     *                    QUAL Function POPUPHIDE
396:     *
397:     *
398:     *
399:
400:     FUNCTION popuphide
401:     PARAMETERS W
402:     RELEASE WINDOW w_popnote
403: <═══RETURN W
404:
405:     *-------------------------------------
406:     *
407:     * This procedure validate the object to have access to delete or not.
408:     *
409:     *-------------------------------------
410:
411:     *
412:     *
413:     *                    QUAL Function VALIDOBJ
414:     *
415:     *
416:     *
417:
418:     FUNCTION validobj
419:     PRIVATE msel,mvalid
420:     mvalid = .F.
421:     msel = SELECT()
422:     SELECT quals
423:     SEEK mquals[mq,4]
424:    ┌IF FOUND()
425:    │   IF EMPTY(rules) AND EMPTY(ruleso)
426:    │       mvalid = .T.
427:    │   ELSE
428:    │       msg = ALLTRIM(STRTRAN(rules,"|",","))
429:    │       IF !EMPTY(ruleso)
430:    │           msg = msg + IIF(!EMPTY(msg), ",","") + ALLTRIM(STRTRAN(rules
=> o,"|",","))
431:    │       ENDIF
432:    │       IF LEN(msg) > 120
433:    │           msg = SUBSTR(msg,1,120)
434:    │           msg = SUBSTR(msg,1,RAT(",",msg)-1) + ", etc"
435:    │       ENDIF
436:    │       mvalid = .F.
437:    │       =errmsg("This object was used by rules: " + msg + ".  Delete wa
=> s not allows!!.",2)
438:    │   ENDIF
439:    └ENDIF
440:     SELECT (msel)
441: <═══RETURN mvalid
442:
443:
444:     *
445:     *
446:     *                    QUAL Function QUALDEL
447:     *
448:     *
449:     *
450:
451:     FUNCTION qualdel
452:     PRIVATE msel
453:     msel = SELECT()
454:     SELECT quals
455:     SEEK mquals[mq,4]
456:    ┌IF FOUND()
457:    │   = popupshow("deleting...")
458:    │   DELETE
459:    │   PACK
460:    │   = popuphide()
461:    └ENDIF
462:     DO setquals
463:     SELECT (msel)
464: <═══RETURN
465:  *:  EOF: QUAL.ac1
```

```
*
*       08/09/94              GOAL.SPR                  09:39:04
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
=> *       │                                                  │
=> *       │              Windows Window definitions          │
=> *       │                                                  │
=> *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_goal") ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PJX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.SCX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.MNX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PRG" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.FRX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.QPR"
      DEFINE WINDOW w_goal ;
         AT  0.000, 0.000  ;
         SIZE 22.500,48.375 ;
         TITLE "Goal Object Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,128,128,128)
      MOVE WINDOW w_goal CENTER
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
=> *       │                                                  │
=> *       │              GOAL/Windows Screen Layout          │
=> *       │                                                  │
=> *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal SAME
   ELSE
      ACTIVATE WINDOW w_goal NOSHOW
   ENDIF
   @ 20.167,17.000 GET mbuttons ;
      PICTURE "@*HN \<Edit;\<Quit" ;
      SIZE 1.917,6.000,0.750 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kopcm()
   @ 0.583,1.875 GET mg ;
      PICTURE "@&N" ;
      FROM mgoals ;
      SIZE 17.500,44.250 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kopg7()

   IF NOT WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_goal

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF

CASE _DOS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
```

```
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

*
*
*          MS-DOS Window definitions
*
*

*

IF NOT WEXIST("w_goal") ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PJX" ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.SCX" ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.MNX" ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PRG" ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.FRX" ;
      OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.QPR"
   DEFINE WINDOW w_goal ;
      FROM INT((SROW()-19)/2),INT((SCOL()-64)/2) ;
      TO INT((SROW()-19)/2)+18,INT((SCOL()-64)/2)+63 ;
      TITLE "Goal Object Editor" ;
      FLOAT ;
      CLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      COLOR SCHEME 1
ENDIF


*
*
*          GOAL/MS-DOS Screen Layout
*
*

*


#REGION 1
IF WVISIBLE("w_goal")
   ACTIVATE WINDOW w_goal SAME
ELSE
   ACTIVATE WINDOW w_goal NOSHOW
ENDIF
@ 16,17 GET mbuttons ;
   PICTURE "@*HN \<Add;\<OK;\<Cancel" ;
   SIZE 1,8,1 ;
   DEFAULT 1 ;
   VALID _qsf0koptk()
@ 1,1 GET mg ;
   PICTURE "@&N" ;
   FROM mgoals ;
   SIZE 14,60 ;
   DEFAULT 1 ;
   VALID _qsf0kopxh() ;
   COLOR SCHEME 2

IF NOT WVISIBLE("w_goal")
   ACTIVATE WINDOW w_goal
ENDIF

READ CYCLE MODAL

RELEASE WINDOW w_goal

#REGION 0
IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


ENDCASE


*
*
*   _QSFOKOPCM          mbuttons VALID
*
*   Function Origin:
*
*   From Platform:      Windows
*   From Screen:        GOAL,     Record Number:    2
*   Variable:           mbuttons
*   Called By:          VALID Clause
*   Object Type:        Push Button
*   Snippet Number:     1
*
*
*
FUNCTION _qsf0kopcm     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && edit
   IF !EMPTY(mg)
      DO edgoal
   ENDIF
CASE mbuttons = 2   && quit
   mok = .F.
   CLEAR READ
ENDCASE
SHOW GETS


*
*
*   _QSFOKOPG7          mg VALID
*
*   Function Origin:
*
*   From Platform:      Windows
*   From Screen:        GOAL,     Record Number:    3
*   Variable:           mg
*   Called By:          VALID Clause
*   Object Type:        List
*   Snippet Number:     2
*
*
*
FUNCTION _qsf0kopg7     &&  mg VALID
#REGION 1
```

```
245:    DO edgoal
246:
247:    *
248:    *
249:    *       _QSF0KOPTK          mbuttons VALID
250:    *
251:    *       Function Origin:
252:    *
253:    *       From Platform:      MS-DOS
254:    *       From Screen:        GOAL,     Record Number:    6
255:    *       Variable:           mbuttons
256:    *       Called By:          VALID Clause
257:    *       Object Type:        Push Button
258:    *       Snippet Number:     3
259:    *
260:    *
261:    *
262:    FUNCTION _qsf0koptk     &&  mbuttons VALID
263:    #REGION 1
264:    DO CASE
265:    CASE mbuttons = 1
266:       DEACTIVATE WINDOW w_goal
267:       DO object.spr WITH "G"
268:       ACTIVATE WINDOW w_goal
269:       DO setgoals
270:    CASE mbuttons = 2   && ok
271:       mok = .T.
272:       CLEAR READ
273:    CASE mbuttons = 3   && cancel
274:       mok = .F.
275:       CLEAR READ
276:    ENDCASE
277:    SHOW GETS
278:
279:
280:    *
281:    *
282:    *       _QSF0KOPXH          mg VALID
283:    *
284:    *       Function Origin:
285:    *
286:    *       From Platform:      MS-DOS
287:    *       From Screen:        GOAL,     Record Number:    7
288:    *       Variable:           mg
289:    *       Called By:          VALID Clause
290:    *       Object Type:        List
291:    *       Snippet Number:     4
292:    *
293:    *
294:    *
295:    FUNCTION _qsf0kopxh     &&  mg VALID
296:    #REGION 1
297:    DO edgoal
298: *:  EOF: GOAL.ac1
```

```
*
*       08/09/94            DISPLAY.SPR                09:39:07
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*
*
*
DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              Windows Window definitions          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_displ") ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.PJX" ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.SCX" ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.MNX" ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.PRG" ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.FRX" ;
         OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.QPR"
      DEFINE WINDOW w_displ ;
         AT  0.000, 0.000  ;
         SIZE 21.250,48.500 ;
         TITLE "Display Object Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,128,128,128)
      MOVE WINDOW w_displ CENTER
   ENDIF


   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              DISPLAY/Windows Screen Layout       │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_displ")
      ACTIVATE WINDOW w_displ SAME
   ELSE
      ACTIVATE WINDOW w_displ NOSHOW
   ENDIF
   @ 18.833,16.000 GET mbuttons ;
      PICTURE "@*HT \<Edit;\<Quit" ;
      SIZE 1.917,6.000,0.750 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0korj1()
   @ 0.667,1.750 GET md ;
      PICTURE "@&N" ;
      FROM mdispl ;
      SIZE 16.333,44.250 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kormj()

   IF NOT WVISIBLE("w_displ")
      ACTIVATE WINDOW w_displ
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_displ

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


CASE _DOS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
```

```
ENDIF
  m.compstat = SET("COMPATIBLE")
  SET COMPATIBLE FOXPLUS

  *
  *
  *          MS-DOS Window definitions
  *
  *
  *

IF NOT WEXIST("w_displ") ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.PJX" ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.SCX" ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.MNX" ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.PRG" ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.FRX" ;
      OR UPPER(WTITLE("W_DISPL")) == "W_DISPL.QPR"
   DEFINE WINDOW w_displ ;
      FROM INT((SROW()-18)/2),INT((SCOL()-64)/2) ;
      TO INT((SROW()-18)/2)+17,INT((SCOL()-64)/2)+63 ;
      TITLE "Display Object Editor" ;
      FLOAT ;
      CLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      COLOR SCHEME 1
ENDIF


  *
  *
  *          DISPLAY/MS-DOS Screen Layout
  *
  *
  *

  #REGION 1
IF WVISIBLE("w_displ")
    ACTIVATE WINDOW w_displ SAME
ELSE
    ACTIVATE WINDOW w_displ NOSHOW
ENDIF
  @ 15,29 GET mbuttons ;
    PICTURE "@*HT OK;Cancel" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
    VALID _qsf0korzx()
  @ 15,11 GET mbbutton ;
    PICTURE "@*HN Add;Delete" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
    VALID _qsf0kos30()
  @ 0,1 GET md ;
    PICTURE "@&N" ;
    FROM mdispl ;
    SIZE 14,60 ;
    DEFAULT 1 ;
    VALID _qsf0kos6a() ;
    COLOR SCHEME 2

  IF NOT WVISIBLE("w_displ")
    ACTIVATE WINDOW w_displ
  ENDIF

  READ CYCLE MODAL

  RELEASE WINDOW w_displ

  #REGION 0
  IF m.talkstat = "ON"
    SET TALK ON
  ENDIF
  IF m.compstat = "ON"
    SET COMPATIBLE ON
  ENDIF


ENDCASE


  *
  *
  *    _QSFOKORJ1            mbuttons VALID
  *
  *    Function Origin:
  *
  *    From Platform:        Windows
  *    From Screen:          DISPLAY,     Record Number:   2
  *    Variable:             mbuttons
  *    Called By:            VALID Clause
  *    Object Type:          Push Button
  *    Snippet Number:       1
  *
  *
  *
  FUNCTION _qsf0korj1     &&  mbuttons VALID
  #REGION 1
DO CASE
CASE mbuttons = 1   && Edit
  IF !EMPTY(md)
    DO eddisplay
  ENDIF
  mok = .T.
CASE mbuttons = 2   && Quit
  mok = .F.
  CLEAR READ
ENDCASE

  *
  *
  *    _QSFOKORMJ            md VALID
  *
  *    Function Origin:
  *
  *    From Platform:        Windows
  *    From Screen:          DISPLAY,     Record Number:   3
  *    Variable:             md
  *    Called By:            VALID Clause
  *    Object Type:          List
  *    Snippet Number:       2
  *
```

```
*
*
FUNCTION _qsf0kormj     &&  md VALID
#REGION 1
DO eddisplay

*
*
*       _QSF0KORZX          mbuttons VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        DISPLAY,     Record Number:    6
*       Variable:           mbuttons
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     3
*
*
*
FUNCTION _qsf0korzx     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && ok
   mok = .T.
CASE mbuttons = 2   && cancel
   mok = .F.
ENDCASE

*
*
*       _QSF0KOS30          mbbutton VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        DISPLAY,     Record Number:    7
*       Variable:           mbbutton
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     4
*
*
*
FUNCTION _qsf0kos30     &&  mbbutton VALID
#REGION 1
DO CASE
CASE mbbutton = 1
   APPEND BLANK
   REPLACE area WITH area.area
CASE mbbutton = 2
ENDCASE
SHOW GETS


*
*
*       _QSF0KOS6A          md VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        DISPLAY,     Record Number:    8
*       Variable:           md
*       Called By:          VALID Clause
*       Object Type:        List
*       Snippet Number:     5
*
*
*
FUNCTION _qsf0kos6a     &&  md VALID
#REGION 1
DO eddisplay
*:  EOF: DISPLAY.ac1
```

```
*
*      08/09/94              DISEASE.SPR                  09:39:10
*
*      Author's Name
*
*      Copyright (c) 1994 Company Name
*      Address
*      City,     Zip
*
*      Description:
*      This program was automatically generated by GENSCRN.
*
*

PARAMETERS m.adding
DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *     ┌──────────────────────────────────────────┐
   *     │                                          │
   *     │          Windows Window definitions      │
   *     │                                          │
   *     └──────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_disease") ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.PJX" ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.SCX" ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.MNX" ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.PRG" ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.FRX" ;
         OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.QPR"
      DEFINE WINDOW w_disease ;
         AT  0.000, 0.000  ;
         SIZE 28.333,66.500 ;
         TITLE "Disease Object Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,0,255,255)
      MOVE WINDOW w_disease CENTER
   ENDIF


   *     ┌──────────────────────────────────────────┐
   *     │                                          │
   *     │        DISEASE/Windows Setup Code - SECTION 2
   *     │                                          │
   *     └──────────────────────────────────────────┘
   *

   #REGION 1
   PRIVATE msel
   msel = SELECT()
   SELECT disease
   SCATTER MEMVAR MEMO

   *     ┌──────────────────────────────────────────┐
   *     │                                          │
   *     │          DISEASE/Windows Screen Layout   │
   *     │                                          │
   *     └──────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_disease")
      ACTIVATE WINDOW w_disease SAME
   ELSE
      ACTIVATE WINDOW w_disease NOSHOW
   ENDIF
   @ 0.333,15.250 SAY "Name"  ;
      FONT "Terminal", 8
   @ 0.417,2.875 SAY "Id"  ;
      FONT "Terminal", 8
   @ 13.417,2.625 SAY "Treatment"  ;
      FONT "Terminal", 8 ;
      STYLE "T"
   @ 1.833,2.750 SAY "Description:"  ;
      FONT "Terminal", 8 ;
      STYLE "T"
   @ 0.500,5.875 GET m.id ;
      SIZE 1.000,5.000 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      DISABLE
   @ 0.417,20.625 GET m.name ;
      SIZE 1.000,41.125 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      PICTURE "@!" ;
      DISABLE
   @ 3.500,2.875 EDIT m.descript ;
      SIZE 9.250,61.000,0.000 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      SCROLL
```

```
      @ 14.833,2.625 EDIT m.treatment ;
         SIZE 10.417,61.000,0.000 ;
         DEFAULT " " ;
         FONT "Terminal", 8 ;
         SCROLL
      @ 25.917,25.750 GET mbuttons ;
         PICTURE "@*HT \<OK;\<Cancel" ;
         SIZE 1.917,7.500,0.750 ;
         DEFAULT 1 ;
         FONT "Terminal", 8 ;
         VALID _qsf0kou0d()

   ┌IF NOT WVISIBLE("w_disease")
   │  ACTIVATE WINDOW w_disease
   └ENDIF

      READ CYCLE MODAL ;
         WHEN _qsf0kou3p()

      RELEASE WINDOW w_disease

      #REGION 0

      SET readborder &rborder

   ┌IF m.talkstat = "ON"
   │  SET TALK ON
   └ENDIF
   ┌IF m.compstat = "ON"
   │  SET COMPATIBLE ON
   └ENDIF

      *
      *
      *        DISEASE/Windows Cleanup Code
      *
      *
      *

      #REGION 1
      SELECT (msel)

 ─CASE _DOS

      *
      *
      *        DISEASE/MS-DOS Setup Code - SECTION 1
      *
      *
      *

      #REGION 1
    ┌IF PARAMETER() = 0
    │  m.adding = 0
```

```
 └ENDIF

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

 ┌IF SET("TALK") = "ON"
 │  SET TALK OFF
 │  m.talkstat = "ON"
 ├ELSE
 │  m.talkstat = "OFF"
 └ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   *
   *
   *        MS-DOS Window definitions
   *
   *
   *

 ┌IF NOT WEXIST("w_disease") ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.PJX" ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.SCX" ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.MNX" ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.PRG" ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.FRX" ;
 │       OR UPPER(WTITLE("W_DISEASE")) == "W_DISEASE.QPR"
 │    DEFINE WINDOW w_disease ;
 │       FROM INT((SROW()-20)/2),INT((SCOL()-78)/2) ;
 │       TO INT((SROW()-20)/2)+19,INT((SCOL()-78)/2)+77 ;
 │       TITLE "Disease Object Editor" ;
 │       FLOAT ;
 │       CLOSE ;
 │       SHADOW ;
 │       NOMINIMIZE ;
 │       COLOR SCHEME 1
 └ENDIF


   *
   *
   *        DISEASE/MS-DOS Screen Layout
   *
   *
   *

   #REGION 1
 ┌IF WVISIBLE("w_disease")
 │  ACTIVATE WINDOW w_disease SAME
 ├ELSE
 │  ACTIVATE WINDOW w_disease NOSHOW
 └ENDIF
   @ 0,19 GET disease.name ;
      SIZE 1,35 ;
      DEFAULT " " ;
```

```
230:         PICTURE "@!" ;
231:         WHEN m.adding = disease.id ;
232:         DISABLE
233:      @ 0,13 SAY "Name" ;
234:         SIZE 1,4, 0
235:      @ 1,0 EDIT disease.descript ;
236:         SIZE 7,76,0 ;
237:         DEFAULT " " ;
238:         SCROLL
239:      @ 9,0 EDIT disease.treatment ;
240:         SIZE 7,76,0 ;
241:         DEFAULT " " ;
242:         SCROLL
243:      @ 17,29 GET mbuttons ;
244:         PICTURE "@*HN \<OK;\<Cancel" ;
245:         SIZE 1,8,1 ;
246:         DEFAULT 1 ;
247:         VALID _qsf0koumi()
248:      @ 0,0 SAY "Id" ;
249:         SIZE 1,2, 0
250:      @ 0,4 GET disease.id ;
251:         SIZE 1,5 ;
252:         DEFAULT " " ;
253:         DISABLE
254:
255:    IF NOT WVISIBLE("w_disease")
256:       ACTIVATE WINDOW w_disease
257:    ENDIF
258:
259:    READ CYCLE MODAL ;
260:       WHEN _qsf0kouqk()
261:
262:    RELEASE WINDOW w_disease
263:
264:    #REGION 0
265:    IF m.talkstat = "ON"
266:       SET TALK ON
267:    ENDIF
268:    IF m.compstat = "ON"
269:       SET COMPATIBLE ON
270:    ENDIF
271:
272:
273: ENDCASE
274:
275:
276:    *
277:    *
278:    *        _QSF0KOU0D            mbuttons VALID
279:    *
280:    *        Function Origin:
281:    *
282:    *        From Platform:        Windows
283:    *        From Screen:          DISEASE,     Record Number:   10
284:    *        Variable:             mbuttons
285:    *        Called By:            VALID Clause
286:    *        Object Type:          Push Button
287:    *        Snippet Number:       1
288:    *
289:    *
290:    *
291:    FUNCTION _qsf0kou0d     &&  mbuttons VALID
292:    #REGION 1
293: DO CASE
294: CASE mbuttons = 1   && ok
295:    SELECT disease
```



```
296:    GATHER MEMVAR MEMO
297:    mok = .T.
298: CASE mbuttons = 2   && cancel
299:    mok = .F.
300: ENDCASE
301:  SHOW GETS
302:
303:    *
304:    *
305:    *        _QSF0KOU3P            Read Level When
306:    *
307:    *        Function Origin:
308:    *
309:    *
310:    *        From Platform:        Windows
311:    *        From Screen:          DISEASE
312:    *        Called By:            READ Statement
313:    *        Snippet Number:       2
314:    *
315:    *
316:    *
317:    FUNCTION _qsf0kou3p     && Read Level When
318:    *
319:    * When Code from screen: DISEASE
320:    *
321:    #REGION 1
322:    SET TOPIC TO "GOAL"
323:    SHOW GET m.name ENABLE
324:     CUROBJ = OBJNUM(m.name)
325:    SHOW GETS
326:
327:
328:    *
329:    *
330:    *        _QSF0KOUMI            mbuttons VALID
331:    *
332:    *        Function Origin:
333:    *
334:    *        From Platform:        MS-DOS
335:    *        From Screen:          DISEASE,     Record Number:   17
336:    *        Variable:             mbuttons
337:    *        Called By:            VALID Clause
338:    *        Object Type:          Push Button
339:    *        Snippet Number:       3
340:    *
341:    *
342:    *
343:    FUNCTION _qsf0koumi     &&  mbuttons VALID
344:    #REGION 1
345: DO CASE
346: CASE mbuttons = 1   && ok
347:    mok = .T.
348:    CLEAR READ
349: CASE mbuttons = 2   && cancel
350:    mok = .F.
351:    CLEAR READ
352: ENDCASE
353:  SHOW GETS
354:
355:    *
356:    *
357:    *        _QSF0KOUQK            Read Level When
358:    *
359:    *        Function Origin:
360:    *
361:    *
```

```
362:     *          From Platform:       MS-DOS
363:     *          From Screen:         DISEASE
364:     *          Called By:           READ Statement
365:     *          Snippet Number:      4
366:     *
367:     *
368:     *
369:     FUNCTION _qsf0kouqk     && Read Level When
370:     *
371:     * When Code from screen: DISEASE
372:     *
373:     #REGION 1
374:     SET TOPIC TO "GOAL"
375:     SHOW GET disease.name ENABLE
376:     _CUROBJ = OBJNUM(disease.name)
377:     SHOW GETS
378:
379:
380:
381:     *
382:     *
383:     *          DISEASE/MS-DOS Supporting Procedures and Functions
384:     *
385:     *
386:     *
387:
388:     #REGION 1
389: *:  EOF: DISEASE.ac1
```

```
  1:  *:**************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\KBLDR.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 08/09/94 at  8:37:34
  8:  *:
  9:  *:         Set by: _QSFOKQC6T()       (function in KBLOAD.SPR)
 10:  *:               : _QSFOKQCZG()       (function in KBLOAD.SPR)
 11:  *:
 12:  *:           Uses: FACT.DBF
 13:  *:               : PREMISE.DBF
 14:  *:               : ACTION.DBF
 15:  *:               : RULE.DBF
 16:  *:               : KBHELP.DBF             Alias: HELP
 17:  *:
 18:  *:        Indexes: FACT.IDX
 19:  *:               : PREMISE.IDX
 20:  *:               : ACTION.IDX
 21:  *:               : RULEAREA.IDX
 22:  *:               : SALIENCE.IDX
 23:  *:               : RULE.IDX
 24:  *:
 25:  *:      Documented 15:00:57                                  FoxDoc versio
=> n 3.00a
 26:  *:**************************************************************************
=> *******
 27:      * kbldr.prg
 28:      *--------------------------------------------------------
 29:      *
 30:      *                      Knowledge Base Loader
 31:      *                             for
 32:      *                 Medical Practice Support System
 33:      *
 34:      *                         $Revision: 1.1 $
 35:      *                     $Date: 92/05/04 11:10:24 $
 36:      *                       $Author: RoyWDobbins $
 37:      *
 38:      *--------------------------------------------------------
 39:      PARAMETERS m.new                          && source directory
 40:
 41:      * Load a new knowledge base
 42:
 43:      * Objects from the new knowledge base are appended to the
 44:      * existing knowledge base files, or replace existing objects.
 45:      * Existing rulesets and their associated premise, action clauses
 46:      * are deleted first, before the new ruleset is loaded
 47:      * Other objects may only be edited, not replaced.
 48:
 49:      PRIVATE m.exists                          && true if item exists
 50:      PRIVATE m.unique                          && enumerated item
 51:      PRIVATE m.uord                            && next ordinal value
 52:      PRIVATE mexact                            && set exact on/off
 53:      PRIVATE m.newrecno                                          && stor
=> e current area record number
 54:
 55:      * check all the database exist at m.new directory.
 56:      * If any database was not define will be stop process.
 57:    ┌─IF !FILE(m.new + "dict.dbf")
 58: <──────RETURN
 59:    └─ENDIF
 60:    ┌─IF !FILE(m.new + "val.dbf")
 61: <──────RETURN
 62:    └─ENDIF
 63:    ┌─IF !FILE(m.new + "enum.dbf")
 64: <──────RETURN
 65:    └─ENDIF
 66:    ┌─IF !FILE(m.new + "disease.dbf")
 67: <──────RETURN
 68:    └─ENDIF
 69:    ┌─IF !FILE(m.new + "help.dbf")
 70: <──────RETURN
 71:    └─ENDIF
 72:    ┌─IF !FILE(m.new + "area.dbf")
 73: <──────RETURN
 74:    └─ENDIF
 75:    ┌─IF !FILE(m.new + "fact.dbf")
 76: <──────RETURN
 77:    └─ENDIF
 78:    ┌─IF !FILE(m.new + "premise.dbf")
 79: <──────RETURN
 80:    └─ENDIF
 81:    ┌─IF !FILE(m.new + "action.dbf")
 82: <──────RETURN
 83:    └─ENDIF
 84:    ┌─IF !FILE(m.new + "rule.dbf")
 85: <──────RETURN
 86:    └─ENDIF
 87:    ┌─IF !FILE(m.new + "goals.dbf")
 88: <──────RETURN
 89:    └─ENDIF
 90:    ┌─IF !FILE(m.new + "display.dbf")
 91: <──────RETURN
 92:    └─ENDIF
 93:
 94:      * Open the database
 95:      SELECT 0
 96:      USE fact INDEX fact
 97:      SELECT 0
 98:      USE premise INDEX premise
 99:      SET RELATION TO fact INTO fact
100:      SELECT 0
101:      USE action INDEX action
102:      SELECT 0
103:      USE rule INDEX rulearea,salience,rule
104:      SET RELATION TO premise INTO premise, action INTO action
105:      SELECT area
106:      SET RELATION TO area INTO rule
107:
108:      SELECT area
109:      m.newrecno = RECNO()
110:
111:      *-------------------------------------------------------
112:      *
113:      * update dict/enum/val databases
114:      *
115:      *-------------------------------------------------------
116:      SELECT dict
117:      GOTO BOTTOM
118:      m.qualid = id + 1                         && next available id
119:      SET ORDER TO 2                            && use name key
120:      SELECT 0
121:      USE (m.new + "dict") ALIAS newdict        &&
122:
123:      i = MAX( RECCOUNT(), 1 )
124:      PRIVATE mqid, mquid
125:      DIMENSION mqid[i], mquid[i]
126:
127:    ┌─DO WHILE !EOF()
```

```
128:     SCATTER MEMVAR                          && load new dict element
129:     m.newid = m.id                          && new item id
130:     mqid[ RECNO() ] = m.id
131:     SELECT dict
132:     m.name = UPPER( TRIM( m.name ) )        && new item name
133:     IF LEN( m.name ) > LEN( name )
134:        m.name = SUBSTR( m.name, LEN( name ) )
135:     ENDIF
136:     SEEK m.name
137:     IF FOUND()
138:        * item already exists
139:        * some fields cannot be edited - use existing values
140:        m.datatype = datatype
141:        m.exists = .T.
142:        m.id = id
143:     ELSE
144:        m.exists = .F.
145:        m.id = m.id + m.qualid               && use next available id
146:        APPEND BLANK                         && add a new dict element
147:     ENDIF
148:     GATHER MEMVAR                           && update dict record
149:     m.uid = m.id                            && updated item id
150:     IF m.datatype = "N"
151:        * update numeric type
152:        SELECT 0
153:        USE (m.new + "val") ALIAS newval
154:        LOCATE FOR id = m.newid
155:        SCATTER MEMVAR                       && load new val
156:        SELECT VAL
157:        SEEK m.uid
158:        IF !FOUND()
159:           APPEND BLANK                      && add a new val
160:        ENDIF
161:        m.id = m.uid
162:        GATHER MEMVAR                        && update the val record
163:        SELECT newval
164:        USE
165:     ELSE
166:        * update enumerated types
167:        * You can only append additional enumerated values
168:        SELECT enum
169:        SEEK m.uid
170:        m.exists = FOUND()                   && see if record(s) already
=> there
171:        m.uord = 0
172:        IF m.exists
173:           * next available ordinal value
174:           COUNT WHILE id = m.uid .AND. !EOF() TO m.uord
175:        ENDIF
176:        SELECT 0
177:        USE (m.new + "enum") ALIAS newenum
178:        LOCATE FOR id = m.newid
179:        DO WHILE id = m.newid .AND. !EOF()
180:           SCATTER MEMVAR                    && load new enum
181:           m.unique = .T.
182:           IF m.exists
183:              * existing element, ensure enumeration is new
184:              SELECT enum
185:              SEEK m.uid
186:              mexact = SET("EXACT")
187:              SET EXACT ON                   && check exact matches
188:              DO WHILE id = m.uid .AND. !EOF()
189:                 IF enumerate = m.enumerate
190:                    m.unique = .F.          && already got this
191:                 EXIT
192:              ENDIF
193:              SKIP
194:           ENDDO
195:           IF mexact = "OFF"
196:              SET EXACT OFF
197:           ENDIF
198:           SELECT newenum
199:        ENDIF
200:        IF m.unique
201:           SELECT enum
202:           APPEND BLANK                      && add a new enum
203:           m.id = m.uid
204:           m.uord = m.uord + 1               && assign next ordinal value
205:           m.ord = m.uord
206:           GATHER MEMVAR                     && update enum record
207:           SELECT newenum
208:        ENDIF
209:        SKIP
210:     ENDDO
211:     USE
212:  ENDIF
213:  SELECT newdict
214:  mquid[recno()] = m.uid                     && update id
215:  SKIP
216: ENDDO
217: USE
218: SELECT dict
219: SET ORDER TO 1                             && restore primary index
220:
221: *-----------------------------------------------------
222: *
223: * update disease file
224: *
225: *-----------------------------------------------------
226: SELECT disease
227: GOTO BOTTOM
228: m.newid = id                               && next available id
229: m.newid = MAX(m.newid, 10000)               && starting at 10000
230: m.newid = m.newid + 1
231: SET ORDER TO 2                             && name order
232:
233: SELECT 0
234: USE (m.new + "disease") ALIAS newdis
235:
236: i = MAX( RECCOUNT(), 1 )
237: PRIVATE mgid, mguid
238: DIMENSION mgid[ i ],  mguid[ i ]
239:
240: DO WHILE !EOF()
241:    SCATTER MEMVAR MEMO                     && load new disease
242:    mgid[ RECNO() ] = m.id
243:    SELECT disease
244:    m.name = UPPER( TRIM( m.name ) )        && new item name
245:    IF LEN( m.name ) > LEN( name )
246:       m.name = SUBSTR( m.name, LEN( name ) )
247:    ENDIF
248:    SEEK m.name
249:    IF FOUND()
250:       m.exists = .T.
251:       m.id = id
252:       m.name = name
253:    ELSE
254:       m.exists = .F.
255:       m.id = m.id + m.newid
256:       APPEND BLANK                          && add a new disease
257:    ENDIF
258:    GATHER MEMVAR MEMO                       && update the disease record
```

```
        SELECT newdis
        mguid[recno()] = m.id                   && update id
        SKIP
    ENDDO
    USE
    SELECT disease
    SET ORDER TO 1


    *------------------------------------------------------
    *
    * update help file
    *
    *------------------------------------------------------
    SELECT 0
    USE kbhelp ALIAS HELP
    SELECT 0
    USE (m.new + "help") ALIAS newhelp

    DO WHILE !EOF()
        SCATTER MEMVAR                          && load new help
        m.details = details
        m.topic = UPPER(m.topic)
        SELECT HELP
        LOCATE FOR m.topic = UPPER(TOPIC)
        IF !FOUND()
            APPEND BLANK                        && add a new help
        ENDIF
        GATHER MEMVAR                           && update the help record
        REPLACE details WITH m.details
        SELECT newhelp
        SKIP
    ENDDO
    USE
    SELECT HELP
    USE


    *------------------------------------------------------
    *
    * Get the new knowledge base descriptor
    *
    *------------------------------------------------------
    SELECT 0
    USE (m.new + "area") ALIAS newarea
    SCATTER MEMVAR                              && load the record
    USE

    * Lookup knowledge base name (case insensitive)
    SELECT area
    LOCATE FOR LOWER(TRIM(name)) = LOWER( TRIM( m.name ) )   && lookup by
=>  name
    IF FOUND()
        m.area = area
    ELSE
        * new knowledge base
        GOTO BOTTOM
        m.area = area + 1                       && get new id
        APPEND BLANK
    ENDIF
    m.newrecno = RECNO()
    GATHER MEMVAR                               && import new record

    *------------------------------------------------------
    *
    * delete existing rules in knowledge base
    *
    *------------------------------------------------------
    SELECT rule
    SEEK m.area
    DO WHILE area = m.area .AND. !EOF()
        SELECT premise
        SEEK rule.premise
        DO WHILE clause = rule.premise
            IF EMPTY( premise.op )
                SELECT fact
                DELETE FOR clause = premise.fact   && delete all premise cla
=> uses
            ENDIF
            SELECT premise
            DELETE
            SKIP
        ENDDO
        SELECT action
        DELETE FOR clause = rule.action         && delete all action clauses
        DELETE FOR clause = rule.else           && delete all else clauses
        SELECT rule
        DELETE                                  && delete the rule
        SKIP
    ENDDO

    *------------------------------------------------------
    *
    * prepare fact database
    *
    *------------------------------------------------------
    SELECT fact
    PACK                                        && purge deleted facts
    GOTO BOTTOM
    m.fact = clause                             && get last clause

    * update fact clauses
    SELECT 0
    USE (m.new + "fact")
    COPY TO "tempxxxx"                          && renumber clauses
    USE ("tempxxxx")
    REPLACE ALL clause WITH m.fact + clause
    REPLACE ALL id WITH ;
        IIF(OBJECT = "D", mguid[ ASCAN(mgid, id) ], ;
        mquid[ ASCAN(mqid, id) ] )
    * update any indirect references
    GOTO TOP

    DO WHILE !EOF()
        m.val = VAL
        FOR i = 1 TO 2
            K = AT( "@", m.val, i )
            IF K = 0
                EXIT
            ENDIF
            m.id    =  VAL ( SUBSTR( m.val, K + 1 ) )
            m.newid  =  mquid[ ASCAN( mqid, m.id ) ]
            m.val = STRTRAN(  m.val, LTRIM( STR( m.id ) ), LTRIM( STR( m.ne
=> wid ) ) )
        NEXT
        REPLACE VAL WITH m.val
        SKIP
    ENDDO

    USE
    SELECT fact
    APPEND FROM ("tempxxxx")                    && load new facts
```

```
388:     DELETE FILE "tempxxxx.dbf"              && remove temporary files
389:
390:     *---------------------------------------------------------
391:     *
392:     * prepare premise database
393:     *
394:     *---------------------------------------------------------
395:     SELECT premise
396:     PACK                                    && purge deleted premises
397:     GOTO BOTTOM
398:     m.premise = clause                      && get last clause
399:
400:     * update premises clauses
401:     SELECT 0
402:     USE (m.new + "premise")
403:     COPY TO "tempxxxx"                      && renumber clauses
404:     USE ("tempxxxx")
405:     REPLACE ALL clause WITH m.premise + clause
406:     REPLACE ALL fact WITH m.fact + fact  FOR EMPTY(op)
407:     USE
408:     SELECT premise
409:     APPEND FROM ("tempxxxx")                && load new premises
410:     DELETE FILE "tempxxxx.dbf"              && remove temporary files
411:
412:     *---------------------------------------------------------
413:     *
414:     * prepare action database
415:     *
416:     *---------------------------------------------------------
417:     SELECT action
418:     PACK                                    && purge delete actions
419:     GOTO BOTTOM
420:     m.action = clause                       && get last clause
421:
422:     * update action clauses
423:     SELECT 0
424:     USE (m.new + "action")
425:     COPY TO "tempxxxx"                      && renumber clauses
426:     USE ("tempxxxx")
427:     REPLACE ALL ;
428:        clause WITH m.action + clause
429:     REPLACE ALL id WITH ;
430:        IIF(OBJECT = "D", mguid[ ASCAN(mgid, id) ], ;
431:        mquid[ ASCAN(mqid, id) ] )
432:     * update any indirect references
433:     GOTO TOP
434:
435:     DO WHILE !EOF()
436:        m.val = VAL
437:        FOR i = 1 TO 2
438:           K = AT( "@", m.val, i )
439:           IF K = 0
440:           EXIT
441:           ENDIF
442:           m.id     =  VAL ( SUBSTR( m.val, K + 1 ) )
443:           m.newid  =  mquid[ ASCAN( mqid, m.id ) ]
444:           m.val = STRTRAN(  m.val, LTRIM( STR( m.id ) ), LTRIM( STR( m.ne
=> wid ) ) )
445:        NEXT
446:        REPLACE VAL WITH m.val
447:        SKIP
448:     ENDDO
449:
450:     USE
451:     SELECT action
452:     APPEND FROM ("tempxxxx")                && load new actions
453:     DELETE FILE "tempxxxx.dbf"              && remove temporary files
454:
455:     *---------------------------------------------------------
456:     *
457:     * update rulebase
458:     *
459:     *---------------------------------------------------------
460:     * Count the rules in new knowledge base
461:     SELECT 0
462:     USE (m.new + "rule")
463:     REPLACE area.rules WITH RECCOUNT()      && update record with rule c
=> ount
464:     COPY TO "tempxxxx"
465:     USE ("tempxxxx")
466:     REPLACE ALL ;
467:        premise WITH m.premise + premise, ;
468:        action WITH m.action + action, ;
469:        ELSE WITH IIF( ELSE > 0, m.action + ELSE, 0 ), ;
470:        area WITH m.area
471:     USE
472:
473:     SELECT rule
474:     PACK                                    && purge deleted rules
475:     APPEND FROM ("tempxxxx")                && load the new rules
476:     DELETE FILE "tempxxxx.dbf"              && remove temporary files
477:     IF FILE("tempxxxx.fpt")
478:        DELETE FILE "tempxxxx.fpt"
479:     ENDIF
480:
481:     *---------------------------------------------------------
482:     *
483:     * prepare goals
484:     *
485:     *---------------------------------------------------------
486:     SELECT 0
487:     USE (m.new + "goals")
488:     COPY TO "tempxxxx"
489:     USE ("tempxxxx")
490:     REPLACE ALL id WITH ;
491:        IIF( OBJECT = "D",  mguid[ ASCAN( mgid, id ) ], ;
492:        mquid[ ASCAN( mqid, id ) ] )
493:     REPLACE ALL area WITH m.area
494:     USE
495:     SELECT goals
496:     DELETE FOR area = m.area
497:     PACK
498:     APPEND FROM ("tempxxxx")
499:     DELETE FILE "tempxxxx.dbf"
500:
501:     *---------------------------------------------------------
502:     *
503:     * prepare display list
504:     *
505:     *---------------------------------------------------------
506:     SELECT 0
507:     USE (m.new + "display")
508:     COPY TO "tempxxxx"
509:     USE ("tempxxxx")
510:     REPLACE ALL id WITH ;
511:        IIF( OBJECT = "D",  mguid[ ASCAN( mgid, id ) ], ;
512:        mquid[ ASCAN( mqid, id ) ] )
513:     REPLACE ALL area WITH m.area
514:     USE
515:     SELECT DISPLAY
516:     DELETE FOR area = m.area
517:     PACK
```

```
518:     APPEND FROM ("tempxxxx")
519:     DELETE FILE "tempxxxx.dbf"
520:
521:     *-------------------------------------------------------
522:     *
523:     * prepare qualifiers
524:     *
525:     *-------------------------------------------------------
526:     SELECT quals
527:     DELETE FOR area = m.area
528:     PACK
529:     COPY STRUCTURE TO (m.new + "quals")
530:     SELECT 0
531:     USE (m.new + "quals") ALIAS newquals
532:     INDEX ON id TO (m.new + "quals")
533:     SET INDEX TO (m.new + "quals")
534:
535:     *-------------------------------------------------------
536:     *
537:     * generate rule ==> qualifier cross-references
538:     *
539:     *-------------------------------------------------------
540:     SELECT rule
541:     SEEK m.area                              && set filters
542:
543:     DO WHILE area = m.area .AND. !EOF()      && process the rulebase
544:        * process rule input cross-references
545:        SELECT premise                        && premise clauses for this r
=> ule
546:        SEEK rule.premise
547:        DO WHILE clause = rule.premise .AND. !EOF()
548:           IF EMPTY( premise.op )
549:              SELECT fact                     && clause is a fact
550:              SEEK premise.fact
551:              DO WHILE clause = premise.fact .AND. !EOF()
552:                 SELECT newquals
553:                 SEEK fact.id
554:                 IF !FOUND()
555:                    APPEND BLANK                 && define new qualifier ob
=> ject
556:                    REPLACE id WITH fact.id, OBJECT WITH fact.object
557:                 ENDIF
558:                 REPLACE area WITH m.area
559:                 m.rules = rules
560:                 m.s = ALLTRIM( STR( rule.rule ) )
561:                 * check for duplicates
562:                 i = AT( m.s, m.rules )
563:                 IF i > 0
564:                    IF VAL( SUBSTR( m.rules,i ) )  =  rule.rule
565:                       * skip duplicate entry
566:                       m.s = ""
567:                    ENDIF
568:                 ENDIF
569:                 IF !EMPTY( m.s )
570:                    m.rules = m.rules + IIF( !EMPTY( m.rules ), "|", "" )
=> + m.s
571:                    REPLACE rules WITH m.rules
572:                 ENDIF
573:                 m.quals = rule.quals
574:                 m.s = ALLTRIM( STR( fact.id ) )
575:                 * check for duplicates
576:                 i = AT(m.s, m.quals)
577:                 IF i > 0
578:                    IF VAL( SUBSTR( m.quals,i ) )  =  fact.id
579:                       * skip duplicate entry
580:                       m.s = ""
581:                    ENDIF
582:                 ENDIF
583:                 IF !EMPTY( m.s )
584:                    m.quals = m.quals + IIF( !EMPTY( m.quals ), "|", "" )
=> + m.s
585:                    REPLACE rule.quals WITH m.quals
586:                 ENDIF
587:                 SELECT fact
588:                 SKIP
589:              ENDDO
590:           ENDIF
591:           SELECT premise
592:           SKIP
593:        ENDDO
594:
595:        * process rule output cross-references
596:        SELECT action                         && rule actions
597:        SEEK rule.action
598:        DO WHILE clause = rule.action .AND. !EOF()
599:           SELECT newquals
600:           SEEK action.id
601:           IF !FOUND()
602:              APPEND BLANK                    && define qualifier/goal
603:              REPLACE id WITH action.id, OBJECT WITH action.object
604:           ENDIF
605:           REPLACE area WITH m.area
606:           m.rules = ruleso
607:           m.s = ALLTRIM( STR( rule.rule ) )
608:           * check for duplicates
609:           i = AT(m.s, m.rules)
610:           IF i > 0
611:              IF VAL( SUBSTR( m.rules,i ) )  =  rule.rule
612:                 * skip duplicate entry
613:                 m.s = ""
614:              ENDIF
615:           ENDIF
616:           IF !EMPTY( m.s )
617:              m.rules = m.rules + IIF( !EMPTY( m.rules ), "|", "" ) + m.s
618:              REPLACE ruleso WITH m.rules
619:           ENDIF
620:           m.goals = rule.goals
621:           m.s = ALLTRIM( STR( action.id ) )
622:           * check for duplicates
623:           i = AT( m.s, m.goals )
624:           IF i > 0
625:              IF VAL( SUBSTR( m.goals,i ) )  =  action.id
626:                 * skip duplicate entry
627:                 m.s = ""
628:              ENDIF
629:           ENDIF
630:           IF !EMPTY( m.s )
631:              m.goals = m.goals + IIF( !EMPTY( m.goals ), "|", "" ) + m.s
632:              REPLACE rule.goals WITH m.goals
633:           ENDIF
634:           SELECT action
635:           SKIP
636:        ENDDO
637:
638:        SELECT rule                           && select another rule
639:        SKIP
640:     ENDDO
641:
642:     * copy qualifier cross-references
643:     SELECT newquals
644:     USE
645:     SELECT quals
```

```
646:     APPEND FROM (m.new + "quals")
647:     DELETE FILE (m.new + "quals.dbf")
648:     DELETE FILE (m.new + "quals.idx")
649:     DELETE FILE (m.new + "quals.fpt")
650:     DELETE FILE (m.new + "area.dbf")
651:     DELETE FILE (m.new + "rule.dbf")
652:     DELETE FILE (m.new + "rule.dbt")
653:     DELETE FILE (m.new + "premise.dbf")
654:     DELETE FILE (m.new + "fact.dbf")
655:     DELETE FILE (m.new + "fact.dbt")
656:     DELETE FILE (m.new + "action.dbf")
657:     DELETE FILE (m.new + "action.dbt")
658:     DELETE FILE (m.new + "disease.dbf")
659:     DELETE FILE (m.new + "disease.dbt")
660:     DELETE FILE (m.new + "dict.dbf")
661:     DELETE FILE (m.new + "val.dbf")
662:     DELETE FILE (m.new + "enum.dbf")
663:     DELETE FILE (m.new + "goals.dbf")
664:     DELETE FILE (m.new + "display.dbf")
665:     DELETE FILE (m.new + "help.dbf")
666:     DELETE FILE (m.new + "help.dbt")
667:
668:     SELECT area
669:     GOTO m.newrecno
670:
671:     * close unuse database
672:     SELECT fact
673:     USE
674:     SELECT premise
675:     USE
676:     SELECT action
677:     USE
678:     SELECT rule
679:     USE
680:
681: <==RETURN
682:
683:  *:  EOF: KBLDR.act
```

```
*
*   08/09/94          ACTION.SPR          09:39:14
*
*   Author's Name
*
*   Copyright (c) 1994 Company Name
*   Address
*   City,     Zip
*
*   Description:
*   This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *        ┌──────────────────────────────────────────────────┐
   *        │                                                  │
   *        │              Windows Window definitions          │
   *        │                                                  │
   *        └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_act") ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.PJX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.SCX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.MNX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.PRG" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.FRX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.QPR"
      DEFINE WINDOW w_act ;
         AT  0.000, 0.000  ;
         SIZE 14.500,57.250 ;
         TITLE "Action [THEN] Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,0,255,255)
      MOVE WINDOW w_act CENTER
   ENDIF

   *        ┌──────────────────────────────────────────────────┐
   *        │                                                  │
   *        │          ACTION/Windows Setup Code - SECTION 2   │
   *        │                                                  │
   *        └──────────────────────────────────────────────────┘
   *

   #REGION 1
   EXTERNAL ARRAY act
   PRIVATE mselect, mok
   mselect = SELECT()
   SELECT action
   IF EOF()
      = errmsg("No < THEN > Statment !!!",1)
      RETURN
   ENDIF


   *        ┌──────────────────────────────────────────────────┐
   *        │                                                  │
   *        │             ACTION/Windows Screen Layout         │
   *        │                                                  │
   *        └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_act")
      ACTIVATE WINDOW w_act SAME
   ELSE
      ACTIVATE WINDOW w_act NOSHOW
   ENDIF
   @ 3.000,47.500 GET mbuttons ;
      PICTURE "@*VN \<Edit;\<Quit" ;
      SIZE 3.000,7.500,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kox5v()
   @ 1.750,2.500 GET ma ;
      PICTURE "@&N" ;
      FROM act ;
      SIZE 10.500,43.375 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kox9d()

   IF NOT WVISIBLE("w_act")
      ACTIVATE WINDOW w_act
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_act
```

```
118:     #REGION 0
119:
120:     SET readborder &rborder
121:
122:     IF m.talkstat = "ON"
123:        SET TALK ON
124:     ENDIF
125:     IF m.compstat = "ON"
126:        SET COMPATIBLE ON
127:     ENDIF
128:
129:
130:     *        ┌──────────────────────────────────────────────────────┐
131:     *        │                                                      │
132:     *        │              ACTION/Windows Cleanup Code             │
133:     *        │                                                      │
134:     *        └──────────────────────────────────────────────────────┘
135:     *
136:
137:     #REGION 1
138:     SELECT (mselect)
139: <-- RETURN
140:
141:
142:
143: CASE _DOS
144:
145:
146:     #REGION 0
147:     REGIONAL m.currarea, m.talkstat, m.compstat
148:
149:     IF SET("TALK") = "ON"
150:        SET TALK OFF
151:        m.talkstat = "ON"
152:     ELSE
153:        m.talkstat = "OFF"
154:     ENDIF
155:     m.compstat = SET("COMPATIBLE")
156:     SET COMPATIBLE FOXPLUS
157:
158:     *        ┌──────────────────────────────────────────────────────┐
159:     *        │                                                      │
160:     *        │              MS-DOS Window definitions               │
161:     *        │                                                      │
162:     *        └──────────────────────────────────────────────────────┘
163:     *
164:
165:     IF NOT WEXIST("w_act") ;
166:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.PJX" ;
167:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.SCX" ;
168:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.MNX" ;
169:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.PRG" ;
170:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.FRX" ;
171:           OR UPPER(WTITLE("W_ACT")) == "W_ACT.QPR"
172:        DEFINE WINDOW w_act ;
173:           FROM INT((SROW()-15)/2),INT((SCOL()-76)/2) ;
174:           TO INT((SROW()-15)/2)+14,INT((SCOL()-76)/2)+75 ;
175:           TITLE "Action Editor" ;
176:           FLOAT ;
177:           CLOSE ;
178:           SHADOW ;
179:           NOMINIMIZE ;
180:           COLOR SCHEME 1
181:     ENDIF
182:
183:
184:     *        ┌──────────────────────────────────────────────────────┐
185:     *        │                                                      │
186:     *        │          ACTION/MS-DOS Setup Code - SECTION 2        │
187:     *        │                                                      │
188:     *        └──────────────────────────────────────────────────────┘
189:     *
190:
191:     #REGION 1
192:     EXTERNAL ARRAY act
193:     PRIVATE mselect, mok
194:     mselect = SELECT()
195:     SELECT action
196:     IF EOF()
197:        GOTO BOTTOM
198:        m.clause = IIF(RECCOUNT() = 0, 0, clause) + 1
199:        m.action = addnewact()
200:     ENDIF
201:
202:
203:     *        ┌──────────────────────────────────────────────────────┐
204:     *        │                                                      │
205:     *        │              ACTION/MS-DOS Screen Layout             │
206:     *        │                                                      │
207:     *        └──────────────────────────────────────────────────────┘
208:     *
209:
210:     #REGION 1
211:     IF WVISIBLE("w_act")
212:        ACTIVATE WINDOW w_act SAME
213:     ELSE
214:        ACTIVATE WINDOW w_act NOSHOW
215:     ENDIF
216:     @ 5,64 GET mbuttons ;
217:        PICTURE "@*VT \<Delete;\<OK;\<Cancel" ;
218:        SIZE 1,8,1 ;
219:        DEFAULT 1 ;
220:        VALID _qsf0koxva()
221:     @ 1,64 GET mebutton ;
222:        PICTURE "@*VN \<Edit" ;
223:        SIZE 1,8,1 ;
224:        DEFAULT 1 ;
225:        WHEN ma > 0 ;
226:        VALID _qsf0koy5o()
227:     @ 3,64 GET minsbutton ;
228:        PICTURE "@*VN \<Add" ;
229:        SIZE 1,8,1 ;
```

```
          DEFAULT 1 ;
          VALID _qsf0koy8a()
       @ 1,1 GET ma ;
          PICTURE "@&N" ;
          FROM act ;
          SIZE 10,61 ;
          DEFAULT 1 ;
          VALID _qsf0koycp() ;
          COLOR SCHEME 2

    IF NOT WVISIBLE("w_act")
          ACTIVATE WINDOW w_act
    ENDIF

       READ CYCLE MODAL

       RELEASE WINDOW w_act

       #REGION 0
    IF m.talkstat = "ON"
          SET TALK ON
    ENDIF
    IF m.compstat = "ON"
          SET COMPATIBLE ON
    ENDIF


       *
=>     *
=>     *          ACTION/MS-DOS Cleanup Code
=>     *
=>     *
=>     *
       *

       #REGION 1
       SELECT (mselect)
<------RETURN



ENDCASE


   *
   *
   *        _QSFOKOX5V          mbuttons VALID
   *
   *        Function Origin:
   *
   *        From Platform:      Windows
   *        From Screen:        ACTION,     Record Number:    2
   *        Variable:           mbuttons
   *        Called By:          VALID Clause
   *        Object Type:        Push Button
   *        Snippet Number:     1
   *
   *
   *
FUNCTION _qsf0kox5v     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1    && Edit
   IF ma > 0
      DO edact
   ENDIF
CASE mbuttons = 2    && Quit
   CLEAR READ
ENDCASE

   *
   *
   *        _QSFOKOX9D          ma VALID
   *
   *        Function Origin:
   *
   *        From Platform:      Windows
   *        From Screen:        ACTION,     Record Number:    3
   *        Variable:           ma
   *        Called By:          VALID Clause
   *        Object Type:        List
   *        Snippet Number:     2
   *
   *
   *
FUNCTION _qsf0kox9d     &&  ma VALID
#REGION 1
DO edact

   *
   *
   *        _QSFOKOXVA          mbuttons VALID
   *
   *        Function Origin:
   *
   *        From Platform:      MS-DOS
   *        From Screen:        ACTION,     Record Number:    6
   *        Variable:           mbuttons
   *        Called By:          VALID Clause
   *        Object Type:        Push Button
   *        Snippet Number:     3
   *
   *
   *
FUNCTION _qsf0koxva     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1          && DELETE
   SEEK rule.action
   DO WHILE clause = rule.action
      DELETE
      SKIP
   ENDDO
   PACK
   mok = .F.
CASE mbuttons = 2    && ok
   mok = .T.
CASE mbuttons = 3    && cancel
   mok = .F.
ENDCASE

   *
   *
   *        _QSFOKOY50          mebutton VALID
   *
   *        Function Origin:
   *
   *        From Platform:      MS-DOS
```

```
357:     *         From Screen:          ACTION,     Record Number:    7
358:     *         Variable:             mebutton
359:     *         Called By:            VALID Clause
360:     *         Object Type:          Push Button
361:     *         Snippet Number:       4
362:     *
363:     *
364:     *
365:     FUNCTION _qsf0koy5o     &&  mebutton VALID
366:     #REGION 1
367:     DO edact
368:
369:     *
370:     *
371:     *         _QSF0KOY8A            minsbutton VALID
372:     *
373:     *         Function Origin:
374:     *
375:     *         From Platform:        MS-DOS
376:     *         From Screen:          ACTION,     Record Number:    8
377:     *         Variable:             minsbutton
378:     *         Called By:            VALID Clause
379:     *         Object Type:          Push Button
380:     *         Snippet Number:       5
381:     *
382:     *
383:     *
384:     FUNCTION _qsf0koy8a     &&  minsbutton VALID
385:     #REGION 1
386:     PRIVATE m.sel
387:     m.sel = SELECT()
388:     m.action = rule.action
389:     SELECT action
390:     APPEND BLANK
391:     REPLACE clause WITH m.action
392:     DO clause.spr WITH m.action
393:    ┌IF EMPTY(op) AND EMPTY(OBJECT) AND EMPTY(id)
394:    │   DELETE
395:    │   PACK
396:    └ENDIF
397:     SELECT (m.sel)
398:     DO setact
399:     *show get mbbutton enable
400:     *show gets
401:
402:
403:
404:     *
405:     *
406:     *         _QSF0KOYCP            ma VALID
407:     *
408:     *         Function Origin:
409:     *
410:     *         From Platform:        MS-DOS
411:     *         From Screen:          ACTION,     Record Number:    9
412:     *         Variable:             ma
413:     *         Called By:            VALID Clause
414:     *         Object Type:          List
415:     *         Snippet Number:       6
416:     *
417:     *
418:     *
419:     FUNCTION _qsf0koycp     &&  ma VALID
420:     #REGION 1
421:     DO edact
422:
423:
424:     *
425:     *
426:     *         ACTION/MS-DOS Supporting Procedures and Functions
427:     *
428:     *
429:     *
430:
431:     #REGION 1
432:     PROCEDURE edact
433:     EXTERNAL ARRAY actr
434:     SET TOPIC TO "EDIT"
435:     SELECT "ACTION"
436:    ┌IF actr[ma] > 0
437:    │   GOTO actr[ma]
438:    └ENDIF
439:     m.clause = rule.action
440:     DO clause.spr
441:     DO setact
442:     SHOW GETS
443:     mtopic = ALIAS()
444:     SET TOPIC TO &mtopic
445: <═══RETURN
446:
447:
448:     FUNCTION addnewact
449:     PRIVATE m.sel
450:     m.sel = SELECT()
451:     SELECT action
452:     GOTO BOTTOM
453:     m.action = IIF(EOF(),0,clause) + 1
454:     DO clause.spr WITH m.action
455:     SEEK m.action
456:    ┌IF FOUND()
457:    │   SELECT rule
458:    │   REPLACE action WITH m.action
459:    └ENDIF
460:     SELECT (m.sel)
461:     DO setact
462: <═══RETURN
463:  *:  EOF: ACTION.ac1
```

```
*
*       08/09/94              ACTELSE.SPR                09:39:18
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              Windows Window definitions          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_act") ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.PJX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.SCX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.MNX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.PRG" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.FRX" ;
         OR UPPER(WTITLE("W_ACT")) == "W_ACT.QPR"
      DEFINE WINDOW w_act ;
         AT  0.000, 0.000  ;
         SIZE 14.333,57.000 ;
         TITLE "Action [ELSE] se) Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,0,255,255)
      MOVE WINDOW w_act CENTER
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │         ACTELSE/Windows Setup Code - SECTION 2   │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   EXTERNAL ARRAY actelse
   PRIVATE mselect, mok
   mselect = SELECT()
   SELECT action
   IF EOF()
      =errmsg("No <Else> statement !!!",1)
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              ACTELSE/Windows Screen Layout       │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_act")
      ACTIVATE WINDOW w_act SAME
   ELSE
      ACTIVATE WINDOW w_act NOSHOW
   ENDIF
   @ 4.333,47.750 GET mbuttons ;
      PICTURE "@*VN \<Edit;\<Quit" ;
      SIZE 2.583,7.500,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kp07d()
   @ 2.000,2.375 GET me ;
      PICTURE "@&N" ;
      FROM actelse ;
      SIZE 10.500,43.250 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kp0ar()

   IF NOT WVISIBLE("w_act")
      ACTIVATE WINDOW w_act
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_act

   #REGION 0
```

```
SET readborder &rborder

IF m.talkstat = "ON"
    SET TALK ON
ENDIF
IF m.compstat = "ON"
    SET COMPATIBLE ON
ENDIF


 *
 *
 *       ACTELSE/Windows Cleanup Code
 *
 *
 *

 #REGION 1
 SELECT (mselect)
 RETURN

CASE _DOS


 #REGION 0
 REGIONAL m.currarea, m.talkstat, m.compstat

 IF SET("TALK") = "ON"
     SET TALK OFF
     m.talkstat = "ON"
 ELSE
     m.talkstat = "OFF"
 ENDIF
 m.compstat = SET("COMPATIBLE")
 SET COMPATIBLE FOXPLUS

 *
 *
 *       MS-DOS Window definitions
 *
 *
 *

 IF NOT WEXIST("w_act") ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.PJX" ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.SCX" ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.MNX" ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.PRG" ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.FRX" ;
        OR UPPER(WTITLE("W_ACT")) == "W_ACT.QPR"
    DEFINE WINDOW w_act ;
        FROM INT((SROW()-15)/2),INT((SCOL()-76)/2) ;
        TO INT((SROW()-15)/2)+14,INT((SCOL()-76)/2)+75 ;
        TITLE "Action (Else) Editor" ;
```

```
        FLOAT ;
        CLOSE ;
        SHADOW ;
        NOMINIMIZE ;
        COLOR SCHEME 1
 ENDIF


 *
 *
 *       ACTELSE/MS-DOS Setup Code - SECTION 2
 *
 *
 *

 #REGION 1
 EXTERNAL ARRAY actelse
 PRIVATE mselect, mok
 mselect = SELECT()
 SELECT action
 IF EOF()
    GOTO BOTTOM
    m.clause = IIF(RECCOUNT() = 0, 0, clause) + 1
    m.action = addnewelse()
 ENDIF



 *
 *
 *       ACTELSE/MS-DOS Screen Layout
 *
 *
 *

 #REGION 1
 IF WVISIBLE("w_act")
    ACTIVATE WINDOW w_act SAME
 ELSE
    ACTIVATE WINDOW w_act NOSHOW
 ENDIF
 @ 5,64 GET mbuttons ;
    PICTURE "@*VT \<Delete;\<OK;\<Cancel" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
    VALID _qsf0kp0s3()
 @ 1,64 GET mebutton ;
    PICTURE "@*VN \<Edit" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
    WHEN me > 0 ;
    VALID _qsf0kp0w3()
 @ 3,64 GET minsbutton ;
    PICTURE "@*VN \<Add" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
```

```
      VALID _qsf0kp0yp()
   @ 1,1 GET me ;
      PICTURE "@&N" ;
      FROM actelse ;
      SIZE 10,61 ;
      DEFAULT 1 ;
      VALID _qsf0kp13m() ;
      COLOR SCHEME 2

   IF NOT WVISIBLE("w_act")
      ACTIVATE WINDOW w_act
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_act

   #REGION 0
   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


   *
   *
   *            ACTELSE/MS-DOS Cleanup Code
   *
   *
   *

   #REGION 1
   SELECT (mselect)
   RETURN

ENDCASE


 *
 *
 *    _QSFOKP07D          mbuttons VALID
 *
 *    Function Origin:
 *
 *    From Platform:      Windows
 *    From Screen:        ACTELSE,     Record Number:    2
 *    Variable:           mbuttons
 *    Called By:          VALID Clause
 *    Object Type:        Push Button
 *    Snippet Number:     1
 *
 *
 *
FUNCTION _qsf0kp07d     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && Edit
   IF me > 0
      DO edelse
   ENDIF
CASE mbuttons = 2   && ok
   CLEAR READ
ENDCASE

 *
 *
 *    _QSFOKPOAR          me VALID
 *
 *    Function Origin:
 *
 *    From Platform:      Windows
 *    From Screen:        ACTELSE,     Record Number:    3
 *    Variable:           me
 *    Called By:          VALID Clause
 *    Object Type:        List
 *    Snippet Number:     2
 *
 *
 *
FUNCTION _qsf0kp0ar     &&  me VALID
#REGION 1
DO edelse


 *
 *
 *    _QSFOKPOS3          mbuttons VALID
 *
 *    Function Origin:
 *
 *    From Platform:      MS-DOS
 *    From Screen:        ACTELSE,     Record Number:    6
 *    Variable:           mbuttons
 *    Called By:          VALID Clause
 *    Object Type:        Push Button
 *    Snippet Number:     3
 *
 *
 *
FUNCTION _qsf0kp0s3     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1          && DELETE
   SEEK rule.else
   DO WHILE clause = rule.else
      DELETE
      SKIP
   ENDDO
   PACK
   mok = .F.
CASE mbuttons = 2   && ok
   mok = .T.
CASE mbuttons = 3   && cancel
   mok = .F.
ENDCASE

 *
 *
 *    _QSFOKPOW3          mebutton VALID
 *
 *    Function Origin:
 *
 *    From Platform:      MS-DOS
 *    From Screen:        ACTELSE,     Record Number:    7
```

```
*
*       Variable:           mebutton
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     4
*
*
*
FUNCTION _qsf0kp0w3     &&  mebutton VALID
#REGION 1
DO edelse

*
*
*       _QSF0KP0YP          minsbutton VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        ACTELSE,     Record Number:    8
*       Variable:           minsbutton
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     5
*
*
*
FUNCTION _qsf0kp0yp     &&  minsbutton VALID
#REGION 1
PRIVATE m.sel
m.sel = SELECT()
SELECT action
IF rule.else = 0
    GOTO BOTTOM
    m.clause = IIF(RECCOUNT() = 0, 0, clause) + 1
    = addnewelse()
ELSE
    m.action = rule.else
    APPEND BLANK
    REPLACE clause WITH m.action
    DO clause.spr WITH m.action
    IF EMPTY(op) AND EMPTY(OBJECT) AND EMPTY(id)
        DELETE
        PACK
    ENDIF
ENDIF
SELECT (m.sel)
DO setelse
SHOW GETS


*
*
*       _QSF0KP13M          me VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        ACTELSE,     Record Number:    9
*       Variable:           me
*       Called By:          VALID Clause
*       Object Type:        List
*       Snippet Number:     6
*
*
*
```



```
FUNCTION _qsf0kp13m     &&  me VALID
#REGION 1
DO edelse


*
*
*       ACTELSE/MS-DOS Supporting Procedures and Functions
*
*
*
*


#REGION 1

*
*
*                   ACTELSE Procedure EDELSE
*
*
*
*


PROCEDURE edelse
DO CASE
CASE _DOS
    EXTERNAL ARRAY actelser
    SET TOPIC TO "EDIT"
    SELECT "ACTION"
    IF actelser[me] > 0
        GOTO actelser[me]
    ENDIF
    m.clause = rule.else
    DO clause.spr
    DO setelse
    SHOW GETS
    mtopic = ALIAS()
    SET TOPIC TO &mtopic
<---RETURN

CASE _WINDOWS
    EXTERNAL ARRAY actelser
    SET TOPIC TO "EDIT"
    SELECT "ACTION"
    IF actelser[me] > 0
        GOTO actelser[me]
    ENDIF
    m.clause = rule.else
    DO clause.spr
    DO setelse
    SHOW GETS
    mtopic = ALIAS()
    SET TOPIC TO &mtopic
<---RETURN

ENDCASE

*
*
*                   ACTELSE Procedure ADDELSE
*
*
*
*


PROCEDURE addelse
```

```
489:    DO CASE
490:    CASE _DOS
491:        EXTERNAL ARRAY actelser
492:        SET TOPIC TO "ADD"
493:        SELECT "ACTION"
494:        SCATTER MEMVAR BLANK
495:        m.clause = rule.else
496:        DO clause.spr
497:        DO setelse
498:        SHOW GETS
499:        mtopic = ALIAS()
500:        SET TOPIC TO &mtopic
501: <------RETURN
502:
503:    CASE _WINDOWS
504:        EXTERNAL ARRAY actelser
505:        SET TOPIC TO "ADD"
506:        SELECT "ACTION"
507:        SCATTER MEMVAR BLANK
508:        m.clause = rule.else
509:        DO clause.spr
510:        DO setelse
511:        SHOW GETS
512:        mtopic = ALIAS()
513:        SET TOPIC TO &mtopic
514: <------RETURN
515:
516:
517:    ENDCASE
518:
519:     *
520:     *
521:     *                    ACTELSE Function ADDNEWELSE
522:     *
523:     *
524:     *
525:
526:     FUNCTION addnewelse
527:     PRIVATE m.sel
528:     m.sel = SELECT()
529:     SELECT action
530:     m.action = m.clause
531:     DO clause.spr WITH m.action
532:     SEEK m.action
533:    IF FOUND()
534:        SELECT rule
535:        REPLACE ELSE WITH m.action
536:    ENDIF
537:     SELECT (m.sel)
538:     DO setelse
539: <---RETURN
540:  *:  EOF: ACTELSE.ac1
```

```
*
*       08/09/94              CLAUSE.SPR                09:39:22
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

PARAMETERS m.clause
DO CASE
CASE _WINDOWS

    *       ┌──────────────────────────────────────────────────┐
    *       │                                                  │
    *       │          CLAUSE/Windows Setup Code - SECTION 1   │
    *       │                                                  │
    *       └──────────────────────────────────────────────────┘
    *

    #REGION 1
    PRIVATE mobj, mselect, mtype, mok, mrecno, u
    PRIVATE m.clause, m.op, m.object, m.id, m.tag, m.val, m.text
    DIMENSION mrecno[2]
    m.adding = .F.
    mselect = SELECT()
    *select fact
    m.clause = IIF(PARAMETER() = 0, clause, m.clause)
    IF m.clause <> clause
        SEEK m.clause
        IF !FOUND()
            m.adding = .T.
            m.op = " "
            m.object = ""
            mtype = ""
            = setobject(@m.object, @mtype)
            m.id = 0
            m.val = " "
            m.tag = "F"
        ENDIF
    ELSE
        SCATTER MEMVAR
        IF m.tag = "T"
            m.val = TEXT
        ENDIF
        IF EMPTY(m.object)
            mtype = ""
            =setobject(@m.object, @mtype)
        ENDIF
        IF m.object = "D"
            SELECT disease
            SEEK m.id
            mobj = name
            mtype = "Goal"
        ELSE
            SELECT dict
            SEEK m.id
            mobj = name
            mtype = "Qualifier"
        ENDIF
    ENDIF
    mok = .F.
    SELECT dict
    mrecno[1] = IIF ( EOF(), 0, RECNO() )
    SELECT disease
    mrecno[2] = IIF ( EOF(), 0, RECNO() )
    SELECT (mselect)


    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
    ELSE
        m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    m.rborder = SET("READBORDER")
    SET readborder ON

    *       ┌──────────────────────────────────────────────────┐
    *       │                                                  │
    *       │          Windows Window definitions              │
    *       │                                                  │
    *       └──────────────────────────────────────────────────┘
    *

IF NOT WEXIST("w_clause") ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.PJX" ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.SCX" ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.MNX" ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.PRG" ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.FRX" ;
      OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.QPR"
   DEFINE WINDOW w_clause ;
      AT  0.000, 0.000  ;
      SIZE 23.167,55.375 ;
      TITLE "Clause Editor" ;
      FONT "Terminal", 8 ;
      FLOAT ;
      CLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      COLOR RGB(,,,128,128,0)
   MOVE WINDOW w_clause CENTER
ENDIF


    *       ┌──────────────────────────────────────────────────┐
```

```
*
*                    CLAUSE/Windows Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("w_clause")
   ACTIVATE WINDOW w_clause SAME
ELSE
   ACTIVATE WINDOW w_clause NOSHOW
ENDIF
@ 11.250,2.500 GET minval ;
   PICTURE "@*IVN " ;
   SIZE 1.167,9.000,1.083 ;
   DEFAULT 0 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kp3sz()
@ 4.583,2.625 GET minvbutton ;
   PICTURE "@*IVN " ;
   SIZE 1.250,9.000,1.083 ;
   DEFAULT 0 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kp3x2()
@ 11.333,2.625 SAY "<Value>"  ;
   FONT "Terminal", 8
@ 4.667,2.625 SAY "<Object>"  ;
   FONT "Terminal", 8
@ 1.083,2.625 SAY "Type"  ;
   FONT "Terminal", 8
@ 8.167,2.625 SAY "Operator"  ;
   FONT "Terminal", 8
@ 1.000,43.625 GET mbuttons ;
   PICTURE "@*VT \<OK;\<Cancel" ;
   SIZE 1.917,7.500,1.083 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kp435()
@ 1.083,12.625 GET mtype ;
   PICTURE "@^ Qualifier;Goal" ;
   SIZE 1.500,23.000 ;
   DEFAULT "Qualifier" ;
   FONT "Terminal", 8 ;
   VALID _qsf0kp472()
@ 4.583,12.625 GET mobj ;
   SIZE 1.000,23.125 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   DISABLE
@ 7.917,12.500 GET m.op ;
   PICTURE "@^ <;<=;==;>=;>;!=;=" ;
   SIZE 1.500,5.000 ;
   DEFAULT "<" ;
   FONT "Terminal", 8
@ 12.917,2.750 EDIT m.val ;
   SIZE 9.167,48.625,0.000 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   SCROLL ;
   DISABLE

IF NOT WVISIBLE("w_clause")
   ACTIVATE WINDOW w_clause
ENDIF

READ CYCLE MODAL

RELEASE WINDOW w_clause

#REGION 0

SET readborder &rborder

IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


*
*                    CLAUSE/Windows Cleanup Code
*
*
*

#REGION 1
IF mok
   IF m.adding
      APPEND BLANK
      REPLACE clause WITH m.clause
   ENDIF
   m.val = ALLTRIM(m.val)
   GATHER MEMVAR
   IF m.tag = "T"  .OR. LEN(m.val) > LEN(VAL)
      REPLACE TEXT WITH m.val, TAG WITH "T",VAL WITH " "
   ENDIF
   IF LEFT(mtype,1) = "G"
      m.temp = rule.goals
      m.search = insnewid(@m.temp,m.id)
      IF !m.search
         m.cur = SELECT()
         SELECT rule
         REPLACE goals WITH m.temp
         SELECT (m.cur)
         RELEASE meme m.cur
      ENDIF
   ELSE
      m.temp = rule.quals
      m.search = insnewid(@m.temp,m.id)
      IF !m.search
         m.cur = SELECT()
         SELECT rule
         REPLACE quals WITH m.temp
         SELECT (m.cur)
         RELEASE meme m.cur
      ENDIF
   ENDIF
ELSE
   IF mrecno[1] > 0
```

```
            SELECT dict
            GOTO mrecno[1]
         ENDIF
         IF mrecno[2] > 0
            SELECT disease
            GOTO mrecno[2]
         ENDIF
      ENDIF
      SELECT (mselect)
<     RETURN

   CASE _DOS

      *
      *
      *        CLAUSE/MS-DOS Setup Code - SECTION 1
      *
      *
      *

      #REGION 1
      PRIVATE mobj, mselect, mtype, mok, mrecno, u
      PRIVATE m.clause, m.op, m.object, m.id, m.tag, m.val, m.text
      DIMENSION mrecno[2]
      m.adding = .F.
      mselect = SELECT()
      *select fact
      m.clause = IIF(PARAMETER() = 0, clause, m.clause)
      IF m.clause <> clause
         SEEK m.clause
         IF !FOUND()
            m.adding = .T.
            m.op = " "
            m.object = ""
            mtype = ""
            = setobject(@m.object, @mtype)
            m.id = 0
            m.val = " "
            m.tag = "F"
         ENDIF
      ELSE
         SCATTER MEMVAR
         IF m.tag = "T"
            m.val = TEXT
         ENDIF
         IF EMPTY(m.object)
            mtype = ""
            =setobject(@m.object, @mtype)
         ENDIF
         IF m.object = "D"
            SELECT disease
            SEEK m.id
            mobj = name
            mtype = "Goal"
         ELSE
            SELECT dict
            SEEK m.id
            mobj = name
            mtype = "Qualifier"
         ENDIF
      ENDIF
      mok = .F.
      SELECT dict
      mrecno[1] = IIF ( EOF(), 0, RECNO() )
      SELECT disease
      mrecno[2] = IIF ( EOF(), 0, RECNO() )
      SELECT (mselect)


      #REGION 0
      REGIONAL m.currarea, m.talkstat, m.compstat

      IF SET("TALK") = "ON"
         SET TALK OFF
         m.talkstat = "ON"
      ELSE
         m.talkstat = "OFF"
      ENDIF
      m.compstat = SET("COMPATIBLE")
      SET COMPATIBLE FOXPLUS

      *
      *
      *        MS-DOS Window definitions
      *
      *
      *

      IF NOT WEXIST("w_clause") ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.PJX" ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.SCX" ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.MNX" ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.PRG" ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.FRX" ;
            OR UPPER(WTITLE("W_CLAUSE")) == "W_CLAUSE.QPR"
         DEFINE WINDOW w_clause ;
            FROM INT((SROW()-20)/2),INT((SCOL()-68)/2) ;
            TO INT((SROW()-20)/2)+19,INT((SCOL()-68)/2)+67 ;
            TITLE "Clause Editor" ;
            FLOAT ;
            CLOSE ;
            SHADOW ;
            NOMINIMIZE ;
            COLOR SCHEME 1
      ENDIF


      *
      *
      *        CLAUSE/MS-DOS Screen Layout
      *
      *
      *

      #REGION 1
```

```
362:    IF WVISIBLE("w_clause")
363:       ACTIVATE WINDOW w_clause SAME
364:    ELSE
365:       ACTIVATE WINDOW w_clause NOSHOW
366:    ENDIF
367:    @ 10,1 SAY "<Value>" ;
368:       SIZE 1,7, 0
369:    @ 4,1 SAY "<Object>" ;
370:       SIZE 1,8, 0
371:    @ 11,1 EDIT m.val ;
372:       SIZE 7,64,0 ;
373:       DEFAULT " " ;
374:       SCROLL ;
375:       DISABLE
376:    @ 4,10 GET mobj ;
377:       SIZE 1,37 ;
378:       DEFAULT " " ;
379:       DISABLE
380:    @ 1,1 SAY "Type" ;
381:       SIZE 1,4, 0
382:    @ 6,10 GET m.op ;
383:       PICTURE "@^ <;<=;==;>=;>;!=;=" ;
384:       SIZE 3,6 ;
385:       DEFAULT "<" ;
386:       COLOR SCHEME 1, 2
387:    @ 0,10 GET mtype ;
388:       PICTURE "@^ Qualifier;Goal" ;
389:       SIZE 3,11 ;
390:       DEFAULT "Qualifier" ;
391:       VALID _qsf0kp5hs() ;
392:       COLOR SCHEME 1, 2
393:    @ 10,1 GET minval ;
394:       PICTURE "@*IVN " ;
395:       SIZE 1,7,1 ;
396:       DEFAULT 0 ;
397:       VALID _qsf0kp5lz()
398:    @ 4,1 GET minvbutton ;
399:       PICTURE "@*IVN " ;
400:       SIZE 1,8,1 ;
401:       DEFAULT 0 ;
402:       VALID _qsf0kp5q1()
403:    @ 1,53 GET mbuttons ;
404:       PICTURE "@*VT \<OK;\<Cancel" ;
405:       SIZE 1,8,1 ;
406:       DEFAULT 1 ;
407:       VALID _qsf0kp5uc()
408:    @ 7,1 SAY "Operator" ;
409:       SIZE 1,8, 0
410:
411:    IF NOT WVISIBLE("w_clause")
412:       ACTIVATE WINDOW w_clause
413:    ENDIF
414:
415:    READ CYCLE MODAL
416:
417:    RELEASE WINDOW w_clause
418:
419:    #REGION 0
420:    IF m.talkstat = "ON"
421:       SET TALK ON
422:    ENDIF
423:    IF m.compstat = "ON"
424:       SET COMPATIBLE ON
425:    ENDIF
426:
427:
```

```
428:    *
=>
429:    *
=>
430:    *                    CLAUSE/MS-DOS Cleanup Code
=>
431:    *
=>
432:    *
=>
433:    *
434:
435:    #REGION 1
436:    IF mok
437:       IF m.adding
438:          APPEND BLANK
439:          REPLACE clause WITH m.clause
440:       ENDIF
441:       m.val = ALLTRIM(m.val)
442:       GATHER MEMVAR
443:       IF m.tag = "T"  .OR. LEN(m.val) > LEN(VAL)
444:          REPLACE TEXT WITH m.val, TAG WITH "T",VAL WITH " "
445:       ENDIF
446:       IF LEFT(mtype,1) = "G"
447:          m.temp = rule.goals
448:          m.search = insnewid(@m.temp,m.id)
449:          IF !m.search
450:             m.cur = SELECT()
451:             SELECT rule
452:             REPLACE goals WITH m.temp
453:             SELECT (m.cur)
454:             RELEASE meme m.cur
455:          ENDIF
456:       ELSE
457:          m.temp = rule.quals
458:          m.search = insnewid(@m.temp,m.id)
459:          IF !m.search
460:             m.cur = SELECT()
461:             SELECT rule
462:             REPLACE quals WITH m.temp
463:             SELECT (m.cur)
464:             RELEASE meme m.cur
465:          ENDIF
466:       ENDIF
467:    ELSE
468:       IF mrecno[1] > 0
469:          SELECT dict
470:          GOTO mrecno[1]
471:       ENDIF
472:       IF mrecno[2] > 0
473:          SELECT disease
474:          GOTO mrecno[2]
475:       ENDIF
476:    ENDIF
477:    SELECT (mselect)
478: <- RETURN
479:
480:
481:  ENDCASE
482:
483:
484:    *
485:    *
486:    *     _QSF0KP3SZ          minval VALID
487:    *
488:    *     Function Origin:
```

```
489:     *
490:     *         From Platform:       Windows
491:     *         From Screen:         CLAUSE,     Record Number:    2
492:     *         Variable:            minval
493:     *         Called By:           VALID Clause
494:     *         Snippet Number:      1
495:     *
496:     *
497:     *
498:     FUNCTION _qsf0kp3sz     &&  minval VALID
499:     #REGION 1
500:    ┌DO CASE
501:    ├CASE m.object = "S"
502:    │  ┌DO CASE
503:    │  ├CASE dict.datatype = "N" .OR. dict.datatype = "C"
504:    │  │    SHOW GET m.val ENABLE
505:    │  │     CUROBJ = OBJNUM(m.val)
506:    │  ├CASE dict.datatype = "L" .OR. dict.datatype = "E" .OR. dict.dataty
=> pe = "M"
507:    │  │    SELECT enum
508:    │  │    BROWSE FOR id = dict.id NOEDIT
509:    │  │    m.val = ALLTRIM(enumerate)
510:    │  │    SELECT (mselect)
511:    │  └ENDCASE
512:    ├CASE m.object = "D"
513:    │    SHOW GET m.val ENABLE
514:    │     CUROBJ = OBJNUM(m.val)
515:    └ENDCASE
516:     SHOW GETS
517:
518:
519:     *
520:     *
521:     *         _QSF0KP3X2           minvbutton VALID
522:     *
523:     *         Function Origin:
524:     *
525:     *         From Platform:       Windows
526:     *         From Screen:         CLAUSE,     Record Number:    3
527:     *         Variable:            minvbutton
528:     *         Called By:           VALID Clause
529:     *         Snippet Number:      2
530:     *
531:     *
532:     *
533:     FUNCTION _qsf0kp3x2     &&  minvbutton VALID
534:     #REGION 1
535:     PRIVATE msel
536:     msel = SELECT()
537:    ┌DO CASE
538:    ├CASE m.object = "S"
539:    │    SELECT dict
540:    │  ┌IF datatype = "N" .OR. datatype = "C"
541:    │  │    SHOW GET m.val ENABLE
542:    │  ├ELSE
543:    │  │    SHOW GET m.val DISABLE
544:    │  └ENDIF
545:    ├CASE m.object = "D"
546:    │    SELECT disease
547:    │    SHOW GET m.val ENABLE
548:    └ENDCASE
549:     BROWSE NOEDIT
550:     m.id = id
551:     mobj = name
552:     SELECT (msel)
553:     SHOW GETS
```

```
554:
555:
556:     *
557:     *
558:     *         _QSF0KP435           mbuttons VALID
559:     *
560:     *         Function Origin:
561:     *
562:     *         From Platform:       Windows
563:     *         From Screen:         CLAUSE,     Record Number:    8
564:     *         Variable:            mbuttons
565:     *         Called By:           VALID Clause
566:     *         Object Type:         Push Button
567:     *         Snippet Number:      3
568:     *
569:     *
570:     *
571:     FUNCTION _qsf0kp435     &&  mbuttons VALID
572:     #REGION 1
573:    ┌DO CASE
574:    │    *case mbuttons = 1   && delete
575:    │    *   m.sure = yesno("Sure you want to delete?","YES","NO")
576:    │    *   IF m.sure
577:    │    *      delete
578:    │    *      pack
579:    │    *   ENDIF
580:    │    *   m.id = 0
581:    │    *   m.op = " "
582:    │    *   m.val = " "
583:    │    *   mok = .f.
584:    ├CASE mbuttons = 1   && ok
585:    │    mok = .T.
586:    ├CASE mbuttons = 2   && cancel
587:    │    mok = .F.
588:    └ENDCASE
589:
590:     *
591:     *
592:     *         _QSF0KP472           mtype VALID
593:     *
594:     *         Function Origin:
595:     *
596:     *         From Platform:       Windows
597:     *         From Screen:         CLAUSE,     Record Number:    9
598:     *         Variable:            mtype
599:     *         Called By:           VALID Clause
600:     *         Object Type:         Popup
601:     *         Snippet Number:      4
602:     *
603:     *
604:     *
605:     FUNCTION _qsf0kp472     &&  mtype VALID
606:     #REGION 1
607:     m.object = LEFT( mtype, 1 )
608:     m.object = CHRTRAN( m.object,'GQ','DS' )
609:    ┌IF m.object = "S"
610:    │    SELECT dict
611:    │  ┌IF datatype = "N" .OR. datatype = "C"
612:    │  │    SHOW GET m.val ENABLE
613:    │  ├ELSE
614:    │  │    SHOW GET m.val DISABLE
615:    │  └ENDIF
616:    ├ELSE
617:    │    SELECT disease
618:    │    SHOW GET m.val ENABLE
619:    └ENDIF
```

```
620:     m.id = id
621:     mobj = name
622:     SELECT (mselect)
623:     SHOW GETS
624:
625:
626:     *
627:     *
628:     *          _QSF0KP5HS            mtype VALID
629:     *
630:     *          Function Origin:
631:     *
632:     *          From Platform:        MS-DOS
633:     *          From Screen:          CLAUSE,     Record Number:   21
634:     *          Variable:             mtype
635:     *          Called By:            VALID Clause
636:     *          Object Type:          Popup
637:     *          Snippet Number:       5
638:     *
639:     *
640:     *
641:     FUNCTION _qsf0kp5hs     &&  mtype VALID
642:     #REGION 1
643:     m.object = LEFT( mtype, 1 )
644:     m.object = CHRTRAN( m.object,'GQ','DS' )
645:    IF m.object = "S"
646:       SELECT dict
647:      IF datatype = "N" .OR. datatype = "C"
648:          SHOW GET m.val ENABLE
649:      ELSE
650:          SHOW GET m.val DISABLE
651:      ENDIF
652:    ELSE
653:       SELECT disease
654:       SHOW GET m.val ENABLE
655:    ENDIF
656:     m.id = id
657:     mobj = name
658:     SELECT (mselect)
659:     SHOW GETS
660:
661:
662:     *
663:     *
664:     *          _QSF0KP5LZ            minval VALID
665:     *
666:     *          Function Origin:
667:     *
668:     *          From Platform:        MS-DOS
669:     *          From Screen:          CLAUSE,     Record Number:   22
670:     *          Variable:             minval
671:     *          Called By:            VALID Clause
672:     *          Snippet Number:       6
673:     *
674:     *
675:     *
676:     FUNCTION _qsf0kp5lz     &&  minval VALID
677:     #REGION 1
678:    DO CASE
679:    CASE m.object = "S"
680:       DO CASE
681:       CASE dict.datatype = "N" .OR. dict.datatype = "C"
682:           SHOW GET m.val ENABLE
683:            CUROBJ = OBJNUM(m.val)
684:       CASE dict.datatype = "L" .OR. dict.datatype = "E" .OR. dict.dataty
=> pe = "M"
685:           SELECT enum
686:           BROWSE FOR id = dict.id NOEDIT
687:           m.val = ALLTRIM(enumerate)
688:           SELECT (mselect)
689:       ENDCASE
690:    CASE m.object = "D"
691:       SHOW GET m.val ENABLE
692:        CUROBJ = OBJNUM(m.val)
693:    ENDCASE
694:     SHOW GETS
695:
696:
697:     *
698:     *
699:     *          _QSF0KP5Q1            minvbutton VALID
700:     *
701:     *          Function Origin:
702:     *
703:     *          From Platform:        MS-DOS
704:     *          From Screen:          CLAUSE,     Record Number:   23
705:     *          Variable:             minvbutton
706:     *          Called By:            VALID Clause
707:     *          Snippet Number:       7
708:     *
709:     *
710:     *
711:     FUNCTION _qsf0kp5q1     &&  minvbutton VALID
712:     #REGION 1
713:     PRIVATE msel
714:     msel = SELECT()
715:    DO CASE
716:    CASE m.object = "S"
717:       SELECT dict
718:      IF datatype = "N" .OR. datatype = "C"
719:          SHOW GET m.val ENABLE
720:      ELSE
721:          SHOW GET m.val DISABLE
722:      ENDIF
723:    CASE m.object = "D"
724:       SELECT disease
725:       SHOW GET m.val ENABLE
726:    ENDCASE
727:     BROWSE NOEDIT
728:     m.id = id
729:     mobj = name
730:     SELECT (msel)
731:     SHOW GETS
732:
733:
734:     *
735:     *
736:     *          _QSF0KP5UC            mbuttons VALID
737:     *
738:     *          Function Origin:
739:     *
740:     *          From Platform:        MS-DOS
741:     *          From Screen:          CLAUSE,     Record Number:   24
742:     *          Variable:             mbuttons
743:     *          Called By:            VALID Clause
744:     *          Object Type:          Push Button
745:     *          Snippet Number:       8
746:     *
747:     *
748:     *
749:     FUNCTION _qsf0kp5uc     &&  mbuttons VALID
750:     #REGION 1
```

```
751:    DO CASE
752:       *case mbuttons = 1   && delete
753:       *   m.sure = yesno("Sure you want to delete?","YES","NO")
754:       *   IF m.sure
755:       *      delete
756:       *      pack
757:       *   ENDIF
758:       *   m.id = 0
759:       *   m.op = " "
760:       *   m.val = " "
761:       *   mok = .f.
762:    CASE mbuttons = 1   && ok
763:       mok = .T.
764:    CASE mbuttons = 2   && cancel
765:       mok = .F.
766:    ENDCASE
767:
768:
769:    *
770:    *
771:    *        CLAUSE/MS-DOS Supporting Procedures and Functions
772:    *
773:    *
774:    *
775:
776:    #REGION 1
777:    FUNCTION insnewid
778:    PARAMETER mdata, mitem
779:    PRIVATE m.temp, m.search
780:    m.temp = ALLTRIM(mdata)
781:    m.search = .F.
782:    DO WHILE !EMPTY(m.temp)
783:       m.t1 = dp(m.temp, "|",1)
784:       m.temp = dp(m.temp,"|",2,999)
785:       IF VAL(m.t1) = mitem
786:          m.search = .T.
787:          EXIT
788:       ENDIF
789:    ENDDO
790:    IF !m.search
791:       mdata = ALLTRIM(mdata) + IIF(EMPTY(mdata),"","|") + ALLTRIM(STR(mi
=> tem))
792:    ENDIF
793: <==RETURN m.search
794:
795:    FUNCTION setobject
796:    PARAMETER m.object, mtype
797:    IF "ACTION.DBF" $ DBF()
798:       m.object = "D"
799:       mtype = "Goal"
800:    ELSE
801:       m.object = "S"
802:       mtype = "Qualifier"
803:    ENDIF
804: <==RETURN
805:  *:  EOF: CLAUSE.ac1
```

```
*:*************************************************************************
=> *******
*:
*: Procedure file: C:\CAMD2\KBEDIT\WORKW\DP.PRG
*:         System: Knowledge Base Editor
*:         Author: Hoa L. Ly
*:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
*:  Last modified: 03/15/93 at  9:30:48
*:
*:         Set by: INSNEWID()        (function in CLAUSE.SPR)
*:
*:      Documented 15:01:02                                 FoxDoc versio
=> n 3.00a
*:*************************************************************************
=> *******
    * ===================================================================
=> =================
    * Date: 08/04/92  10:01:15
    * Program Name: DP.PRG
    * Author's Name: Hoa Le Ly
    *
    * Copyright (c) 1992 Company Name: NHRC
    * Departament: Code 22
    * San Diego, CA 92138 - 5122
    * Description: This program emulate $PIECE of MUMPS function.  Which
=> return the
    *    the portion of string which is bounded by the characters in deli
=> miter. If both
    *    expr and expr2 are present, the value returned includes all char
=> acters from
    *     the expr1-1th occurrence of delimiter, up to but not including
=> the expr2th
    *    occurence of delimiter.  If expr2 is not present. it is assumed
=> to have the same
    *     value as expr.  If expr1 is not present. Then its value is assu
=> me to be 1
    * SYNTAX: DP(string, delimiter[,expr[,expr2]])
    * PARAMETER: string: Character expression which character extract fro
=> m
    *                  delimiter: Character to delimiter
    *                      expr: start of number occurrence of delimiter
    *                      expr2: number of occurrence delimiter
    * eg: string = "last, first^age^date of birth"
    *    ?dp(string, "^",1)            ===>  last, first
    *    ?dp(string,"^",1,2)           ===>  last, first^age
    * ===================================================================
=> ================

    PARAMETER ms,mm,mp,mp2
    PRIVATE ALL
  ┌─DO CASE
  ├─CASE PARAMETER() < 2
<─────RETURN ""
  ├─CASE PARAMETER() = 2
  │   mp = 1
  │   mp2 = -1
  ├─CASE PARAMETER() = 3
  │  ┌─IF TYPE("mp") != "N"
<────────RETURN ""
  │  ├─ELSE
  │  │  ┌─IF mp < 1
<───────────RETURN ""
  │  │  └─ENDIF
  │  └─ENDIF
  │   mp2 = -1
  ├─CASE PARAMETER() = 4
  │  ┌─IF TYPE("mp") != "N"
<────────RETURN ""
  │  └─ENDIF
  │  ┌─IF TYPE("mp2") = "N"
  │  │    mp2 = IIF(mp >= mp2, -1, mp2)
  │  ├─ELSE
  │  │    mp2 = -1
  │  └─ENDIF
  └─ENDCASE
  ┌─IF TYPE("ms") != "C"
  │   ms = STR(ms)
  └─ENDIF
   moccurs = OCCURS(mm,ms)
  ┌─IF moccurs = 0                                          && Return ms if
=> mp=1&(ms'[mm)
  │   mstr = IIF(mp = 1, ms, "")                  && Return null if mp>1&(
=> ms'[mm)
<─────RETURN mstr
  └─ENDIF
   mbegin = IIF(mp = 1, 1, (AT(mm,ms,(mp-1))+1))
  ┌─IF mp2 = -1
  │   mend = AT(mm,ms,mp) - 1
  │  ┌─IF mend < 0
  │  │  ┌─IF mbegin > 1
  │  │  │    mend = LEN(ms)
  │  │  ├─ELSE
<───────────RETURN ""
  │  │  └─ENDIF
  │  └─ENDIF
  ├─ELSE
  │   mend = AT(mm,ms,mp2) - 1
  │   mend = IIF(mend<0,LEN(ms),mend)
  └─ENDIF
   mstr = SUBSTR(ms, mbegin, (mend - mbegin +1))
<──RETURN mstr
 *:  EOF: DP.act
```

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\RESTORE.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 06/03/94 at  9:26:30
  8:  *:
  9:  *:         Set by: KBMENU.MPR
 10:  *:
 11:  *:          Calls: RESTORE.SPR
 12:  *:
 13:  *:      Documented 15:01:02                                FoxDoc versio
=> n 3.00a
 14:  *:*************************************************************************
=> *******
 15:      *-------------------------------------------------------------------
=> ----
 16:      * Restore data files
 17:      * Programmer: HLL
 18:      * PROCEDURE restore
 19:      *-------------------------------------------------------------------
=> ---
 20:      PUBLIC mscreen
 21:      PRIVATE MESSAGE
 22:      SET TALK OFF
 23:      SAVE SCREEN TO mscreen
 24:      MESSAGE = "Restore Knowledge Base database"
 25:      SET TOPIC TO "RESTORE"
 26:      DO restore.spr WITH MESSAGE
 27:      RESTORE SCREEN FROM mscreen
 28: <---RETURN
 29:  *:  EOF: RESTORE.act
```

O-B1

```
*
*       ┌──────────────────────────────────────────────────────────┐
*       │ 08/09/94              RULE.SPR                 09:39:29  │
*       │──────────────────────────────────────────────────────────│
*       │                                                          │
*       │ Author's Name                                            │
*       │                                                          │
*       │ Copyright (c) 1994 Company Name                          │
*       │ Address                                                  │
*       │ City,     Zip                                            │
*       │                                                          │
*       │ Description:                                             │
*       │ This program was automatically generated by GENSCRN.     │
*       │                                                          │
*       └──────────────────────────────────────────────────────────┘
*

DO CASE
CASE _WINDOWS


    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
    ELSE
        m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    m.rborder = SET("READBORDER")
    SET readborder ON

    *
    *       Windows Window definitions
    *

    IF NOT WEXIST("w_rule") ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.PJX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.SCX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.MNX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.PRG" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.FRX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.QPR"
        DEFINE WINDOW w_rule ;
            AT  0.000, 0.000  ;
            SIZE 25.333,64.375 ;
            TITLE "Rule Object Editor" ;
            FONT "Terminal", 8 ;
            FLOAT ;
            CLOSE ;
            SHADOW ;
            NOMINIMIZE ;
            COLOR RGB(,,,0,255,255)
        MOVE WINDOW w_rule CENTER
    ENDIF


    *
    *       RULE/Windows Setup Code - SECTION 2
    *

    #REGION 1
    PRIVATE msel
    msel = SELECT()
    SELECT rule
    SET FILTER TO area = area.area
    SCATTER MEMO MEMVAR
    m.adding = .F.
    m.oldrec = RECNO()

    *
    *       RULE/Windows Screen Layout
    *

    #REGION 1
    IF WVISIBLE("w_rule")
        ACTIVATE WINDOW w_rule SAME
    ELSE
        ACTIVATE WINDOW w_rule NOSHOW
    ENDIF
    @ 1.250,1.500 SAY "Rule"  ;
        FONT "Terminal", 8
    @ 1.250,6.625 GET m.rule ;
        SIZE 1.000,6.000 ;
        DEFAULT 0 ;
        FONT "Terminal", 8 ;
        DISABLE
    @ 1.250,15.500 SAY "Salience"  ;
        FONT "Terminal", 8
    @ 1.250,25.125 GET m.salience ;
        SIZE 1.000,5.000 ;
        DEFAULT " " ;
        FONT "Terminal", 8 ;
        VALID m.salience >= 0 .AND. m.salience <=  10000
    @ 22.667,24.000 GET mbuttons ;
        PICTURE "@*HT \<Ok ;\<Cancel" ;
        SIZE 1.917,7.500,0.750 ;
        DEFAULT 1 ;
        FONT "Terminal", 8 ;
        VALID _qsf0kp97y()
    @ 3.333,1.500 SAY "Explanation"  ;
        FONT "Terminal", 8
    @ 13.250,1.250 SAY "Note"  ;
        FONT "Terminal", 8
```

```
    @ 4.750,1.500 EDIT m.explain ;
        SIZE 7.583,60.375,0.000 ;
        DEFAULT " " ;
        FONT "Terminal", 8 ;
        SCROLL
    @ 14.500,1.500 EDIT m.note ;
        SIZE 7.333,60.125,0.000 ;
        DEFAULT " " ;
        FONT "Terminal", 8 ;
        SCROLL

    IF NOT WVISIBLE("w_rule")
        ACTIVATE WINDOW w_rule
    ENDIF

    READ CYCLE MODAL

    RELEASE WINDOW w_rule

    #REGION 0

    SET readborder &rborder

    IF m.talkstat = "ON"
        SET TALK ON
    ENDIF
    IF m.compstat = "ON"
        SET COMPATIBLE ON
    ENDIF


    *
    *
    *           RULE/Windows Cleanup Code
    *
    *
    *

    #REGION 1
    SELECT rule
    SET FILTER TO

CASE _DOS


    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
    ELSE
        m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    *
    *           MS-DOS Window definitions
    *
    *

    IF NOT WEXIST("w_rule") ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.PJX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.SCX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.MNX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.PRG" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.FRX" ;
            OR UPPER(WTITLE("W_RULE")) == "W_RULE.QPR"
        DEFINE WINDOW w_rule ;
            FROM INT((SROW()-20)/2),INT((SCOL()-76)/2) ;
            TO INT((SROW()-20)/2)+19,INT((SCOL()-76)/2)+75 ;
            TITLE "Rule Object Editor" ;
            FLOAT ;
            CLOSE ;
            SHADOW ;
            NOMINIMIZE ;
            COLOR SCHEME 1
    ENDIF


    *
    *
    *           RULE/MS-DOS Setup Code - SECTION 2
    *
    *
    *

    #REGION 1
    PRIVATE msel
    msel = SELECT()
    SELECT rule
    SET FILTER TO area = area.area
    SCATTER MEMO MEMVAR
    m.adding = .F.
    m.oldrec = RECNO()

    *
    *
    *           RULE/MS-DOS Screen Layout
    *
    *
    *

    #REGION 1
    IF WVISIBLE("w_rule")
        ACTIVATE WINDOW w_rule SAME
    ELSE
        ACTIVATE WINDOW w_rule NOSHOW
    ENDIF
```

```
@ 1,1 SAY "Rule" ;
   SIZE 1,4, 0
@ 1,6 GET m.rule ;
   SIZE 1,5 ;
   DEFAULT 0 ;
   DISABLE
@ 1,15 SAY "Salience" ;
   SIZE 1,8, 0
@ 1,24 GET m.salience ;
   SIZE 1,4 ;
   DEFAULT " " ;
   VALID m.salience >= 0 .AND. m.salience <=  10000
@ 17,15 GET mbbutton ;
   PICTURE "@*VN \<Add" ;
   SIZE 1,8,1 ;
   DEFAULT 1 ;
   VALID _qsf0kpa3b()
@ 17,25 GET mbuttons ;
   PICTURE "@*HT \<Delete ;\<Ok ;\<Cancel" ;
   SIZE 1,9,1 ;
   DEFAULT 1 ;
   VALID _qsf0kpa7j()
@ 3,1 SAY "Explanation" ;
   SIZE 1,11, 0
@ 10,1 SAY "Note" ;
   SIZE 1,4, 0
@ 4,1 EDIT m.explain ;
   SIZE 6,72,0 ;
   DEFAULT " " ;
   SCROLL
@ 11,1 EDIT m.note ;
   SIZE 6,72,0 ;
   DEFAULT " " ;
   SCROLL

IF NOT WVISIBLE("w_rule")
   ACTIVATE WINDOW w_rule
ENDIF

READ CYCLE MODAL

RELEASE WINDOW w_rule

#REGION 0
IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


*
*
*          RULE/MS-DOS Cleanup Code
*
*
*

#REGION 1
SELECT rule
   SET FILTER TO

ENDCASE


*
*
*      _QSFOKP97Y            mbuttons VALID
*
*      Function Origin:
*
*      From Platform:        Windows
*      From Screen:          RULE,      Record Number:    6
*      Variable:             mbuttons
*      Called By:            VALID Clause
*      Object Type:          Push Button
*      Snippet Number:       1
*
*
*
FUNCTION _qsf0kp97y     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && ok
   GATHER MEMVAR MEMO
CASE mbuttons = 2   && cancel
   SCATTER MEMO MEMVAR
ENDCASE

*
*
*      _QSFOKPA3B            mbbutton VALID
*
*      Function Origin:
*
*      From Platform:        MS-DOS
*      From Screen:          RULE,      Record Number:    17
*      Variable:             mbbutton
*      Called By:            VALID Clause
*      Object Type:          Push Button
*      Snippet Number:       2
*
*
*
FUNCTION _qsf0kpa3b     &&  mbbutton VALID
#REGION 1
m.adding = .T.
m.oldrec = RECNO()
SET TOPIC TO "ADD"
SCATTER MEMO MEMVAR BLANK
m.area = area.area
GOTO BOTTOM
m.rule = rule + 1
m.order = SET("order")
SET ORDER TO salience
GOTO BOTTOM
m.salience = salience + 10
SET ORDER TO (m.order)
SET TOPIC TO "RULE"
SHOW GET mbbutton disabled
SHOW GETS


*
*
*      _QSFOKPA7J            mbuttons VALID
```

```
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        RULE,      Record Number:   18
*       Variable:           mbuttons
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     3
*
*
*
FUNCTION _qsf0kpa7j     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && delete
   m.del = yesno("Sure You want delete?","YES","NO")
   IF m.del
      m.sel = SELECT()
      SET TOPIC TO "DELETE"
      SELECT premise
      DELETE FOR clause = m.premise
      PACK
      SELECT action
      DELETE FOR clause = m.action
      PACK
      SELECT rule
      DELETE
      PACK
      COUNT TO m.rules
      SELECT area
      REPLACE rules WITH m.rules
      SELECT (m.sel)
   ENDIF
   SCATTER MEMO MEMVAR
   SET TOPIC TO "RULE"
CASE mbuttons = 2   && ok
   IF m.adding
      APPEND BLANK
      GATHER MEMVAR MEMO
      m.recno = RECNO()
      COUNT TO m.rules
      GOTO m.recno
      m.sel = SELECT()
      SELECT area
      REPLACE rules WITH m.rules
      SELECT (m.sel)
   ELSE
      GATHER MEMVAR MEMO
   ENDIF
CASE mbuttons = 3   && cancel
   GOTO m.oldrec
   SCATTER MEMO MEMVAR
ENDCASE
*:  EOF: RULE.ac1
```

```
*
*   08/09/94            TERM.SPR            09:39:33
*
*
*   Author's Name
*
*   Copyright (c) 1994 Company Name
*   Address
*   City,     Zip
*
*   Description:
*   This program was automatically generated by GENSCRN.
*
*
*

DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *
   *
   *              Windows Window definitions
   *
   *
   *

   IF NOT WEXIST("w_term") ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.PJX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.SCX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.MNX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.PRG" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.FRX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.QPR"
      DEFINE WINDOW w_term ;
         AT  0.000, 0.000  ;
         SIZE 19.917,55.250 ;
         TITLE "Term [IF] Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR RGB(,,,0,255,255)
      MOVE WINDOW w_term CENTER
   ENDIF


   *
   *
   *              TERM/Windows Setup Code - SECTION 2
   *
   *
   *

   #REGION 1
   EXTERNAL ARRAY prem, premr, premo, triple
   PRIVATE mselect, mok
   mselect = SELECT()
   SELECT premise
   IF EOF()
      PRIVATE m.flag = .F.
      GOTO BOTTOM
      m.clause = IIF(EOF(),0, clause) + 1
      m.flag = addfact()
      IF m.flag
         PRIVATE m.sel
         m.sel = SELECT()
         SELECT rule
         REPLACE premise WITH m.clause
         SELECT (m.sel)
      ENDIF
   ENDIF


   *
   *
   *              TERM/Windows Screen Layout
   *
   *
   *

   #REGION 1
   IF WVISIBLE("w_term")
      ACTIVATE WINDOW w_term SAME
   ELSE
      ACTIVATE WINDOW w_term NOSHOW
   ENDIF
   @ 1.250,3.375 GET mp ;
      PICTURE "@&N" ;
      FROM prem ;
      SIZE 16.333,36.875 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kpc3r()
   @ 2.833,43.125 GET mbuttons ;
      PICTURE "@*VN \<Edit;\<Quit" ;
      SIZE 4.417,8.000,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kpc6d()
```

```
   IF NOT WVISIBLE("w_term")
      ACTIVATE WINDOW w_term
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_term

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF

   *
   *
   *       TERM/Windows Cleanup Code
   *
   *
   *

   #REGION 1
<==RETURN .T.

CASE _DOS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   *
   *
   *       MS-DOS Window definitions
   *
   *
   *

   IF NOT WEXIST("w_term") ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.PJX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.SCX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.MNX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.PRG" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.FRX" ;
         OR UPPER(WTITLE("W_TERM")) == "W_TERM.QPR"
      DEFINE WINDOW w_term ;
         FROM INT((SROW()-19)/2),INT((SCOL()-76)/2) ;
         TO INT((SROW()-19)/2)+18,INT((SCOL()-76)/2)+75 ;
         TITLE "Term Editor" ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR SCHEME 1
   ENDIF

   *
   *
   *       TERM/MS-DOS Setup Code - SECTION 2
   *
   *
   *

   #REGION 1
   EXTERNAL ARRAY prem, premr, premo, triple
   PRIVATE mselect, mok
   mselect = SELECT()
   SELECT premise
   IF EOF()
      PRIVATE m.flag = .F.
      GOTO BOTTOM
      m.clause = IIF(EOF(),0, clause) + 1
      m.flag = addfact()
      IF m.flag
         PRIVATE m.sel
         m.sel = SELECT()
         SELECT rule
         REPLACE premise WITH m.clause
         SELECT (m.sel)
      ENDIF
   ENDIF

   *
   *
   *       TERM/MS-DOS Screen Layout
   *
   *
   *

   #REGION 1
   IF WVISIBLE("w_term")
      ACTIVATE WINDOW w_term SAME
   ELSE
      ACTIVATE WINDOW w_term NOSHOW
```

```
    ENDIF
  @ 9,63 GET mbuttons ;
     PICTURE "@*VT \<Delete;\<OK;\<Cancel" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0kpcpm()
  @ 3,63 GET mbbutton ;
     PICTURE "@*VN \<AND" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0kpcu9()
  @ 5,63 GET morbutton ;
     PICTURE "@*VN \<OR" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0kpcwu()
  @ 1,63 GET mebutton ;
     PICTURE "@*VN \<Edit" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     WHEN mp > 0 ;
     VALID _qsf0kpczg()
  @ 7,63 GET minsbutton ;
     PICTURE "@*VN \<Insert" ;
     SIZE 1,8,1 ;
     DEFAULT 1 ;
     VALID _qsf0kpd1z()
  @ 0,1 GET mp ;
     PICTURE "@&N" ;
     FROM prem ;
     SIZE 17,59 ;
     DEFAULT 1 ;
     VALID _qsf0kpd51() ;
     COLOR SCHEME 2

  IF NOT WVISIBLE("w_term")
     ACTIVATE WINDOW w_term
  ENDIF

  READ CYCLE MODAL

  RELEASE WINDOW w_term

  #REGION 0
  IF m.talkstat = "ON"
     SET TALK ON
  ENDIF
  IF m.compstat = "ON"
     SET COMPATIBLE ON
  ENDIF


  *
=>
  *
=>
  *            TERM/MS-DOS Cleanup Code
=>
  *
=>
  *
=>
  *

  #REGION 1
<-RETURN .T.
```

```


ENDCASE


*
*    _QSFOKPC3R              mp VALID
*
*    Function Origin:
*
*    From Platform:          Windows
*    From Screen:            TERM,     Record Number:    2
*    Variable:               mp
*    Called By:              VALID Clause
*    Object Type:            List
*    Snippet Number:         1
*
*
*
FUNCTION _qsf0kpc3r     &&  mp VALID
#REGION 1
DO edprem

*
*
*    _QSFOKPC6D              mbuttons VALID
*
*    Function Origin:
*
*    From Platform:          Windows
*    From Screen:            TERM,     Record Number:    3
*    Variable:               mbuttons
*    Called By:              VALID Clause
*    Object Type:            Push Button
*    Snippet Number:         2
*
*
*
FUNCTION _qsf0kpc6d     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1   && Edit
   IF mp > 0
      DO edprem
   ENDIF
   mok = .T.
CASE mbuttons = 2   && cancel
   mok = .F.
   CLEAR READ
ENDCASE

*
*
*    _QSFOKPCPM              mbuttons VALID
*
*    Function Origin:
*
*    From Platform:          MS-DOS
*    From Screen:            TERM,     Record Number:    6
*    Variable:               mbuttons
*    Called By:              VALID Clause
*    Object Type:            Push Button
*    Snippet Number:         3
*
*
```

```
357:    *
358:    FUNCTION _qsf0kpcpm     &&  mbuttons VALID
359:    #REGION 1
360:    DO CASE
361:    CASE mbuttons = 1         && DELETE
362:       SEEK rule.premise
363:       DO WHILE clause = rule.premise
364:          IF EMPTY(premise.op)
365:             SELECT fact
366:             DELETE FOR clause = premise.fact
367:             SELECT premise
368:          ENDIF
369:          DELETE
370:          SKIP
371:       ENDDO
372:       SELECT fact
373:       *  pack                          && wait till the end
374:       SELECT premise
375:       *  pack                          && wait till the end
376:       mok = .F.
377:    CASE mbuttons = 2   && ok
378:       mok = .T.
379:    CASE mbuttons = 3   && cancel
380:       mok = .F.
381:    ENDCASE
382:
383:    *
384:    *
385:    *        _QSF0KPCU9            mbbutton VALID
386:    *
387:    *        Function Origin:
388:    *
389:    *        From Platform:        MS-DOS
390:    *        From Screen:          TERM,     Record Number:    7
391:    *        Variable:             mbbutton
392:    *        Called By:            VALID Clause
393:    *        Object Type:          Push Button
394:    *        Snippet Number:       4
395:    *
396:    *
397:    *
398:    FUNCTION _qsf0kpcu9     &&  mbbutton VALID
399:    #REGION 1
400:    =setjoin("&")
401:
402:    *
403:    *
404:    *        _QSF0KPCWU            morbutton VALID
405:    *
406:    *        Function Origin:
407:    *
408:    *        From Platform:        MS-DOS
409:    *        From Screen:          TERM,     Record Number:    8
410:    *        Variable:             morbutton
411:    *        Called By:            VALID Clause
412:    *        Object Type:          Push Button
413:    *        Snippet Number:       5
414:    *
415:    *
416:    *
417:    FUNCTION _qsf0kpcwu     &&  morbutton VALID
418:    #REGION 1
419:    =setjoin("|")
420:
421:    *
422:    *
```

```
423:    *        _QSF0KPCZG            mebutton VALID
424:    *
425:    *        Function Origin:
426:    *
427:    *        From Platform:        MS-DOS
428:    *        From Screen:          TERM,     Record Number:    9
429:    *        Variable:             mebutton
430:    *        Called By:            VALID Clause
431:    *        Object Type:          Push Button
432:    *        Snippet Number:       6
433:    *
434:    *
435:    *
436:    FUNCTION _qsf0kpczg     &&  mebutton VALID
437:    #REGION 1
438:    DO edprem
439:
440:    *
441:    *
442:    *        _QSF0KPD1Z            minsbutton VALID
443:    *
444:    *        Function Origin:
445:    *
446:    *        From Platform:        MS-DOS
447:    *        From Screen:          TERM,     Record Number:    10
448:    *        Variable:             minsbutton
449:    *        Called By:            VALID Clause
450:    *        Object Type:          Push Button
451:    *        Snippet Number:       7
452:    *
453:    *
454:    *
455:    FUNCTION _qsf0kpd1z     &&  minsbutton VALID
456:    #REGION 1
457:    m.clause = rule.premise
458:    = addfact()
459:    SHOW GET mbbutton ENABLE
460:    SHOW GETS
461:
462:
463:
464:    *
465:    *
466:    *        _QSF0KPD51            mp VALID
467:    *
468:    *        Function Origin:
469:    *
470:    *        From Platform:        MS-DOS
471:    *        From Screen:          TERM,     Record Number:    11
472:    *        Variable:             mp
473:    *        Called By:            VALID Clause
474:    *        Object Type:          List
475:    *        Snippet Number:       8
476:    *
477:    *
478:    *
479:    FUNCTION _qsf0kpd51     &&  mp VALID
480:    #REGION 1
481:    DO edprem
482:
483:
484:    *
485:    *
486:    *        TERM/MS-DOS Supporting Procedures and Functions
487:    *
488:    *
```

```
489:     *
490:
491:     #REGION 1
492:     PROCEDURE edprem
493:     EXTERNAL ARRAY prem, premr, triple
494:     SET TOPIC TO "EDIT"
495:     SELECT fact
496:     K = premr[mp]
497:    ┌IF triple[k,2] > 0
498:    │   SEEK triple[k,2]
499:    ├ELSE
500:    │   * empty fact list; add insert fact !!!
501:    └ENDIF
502:     m.clause = premise.fact
503:     DO clause.spr
504:     SELECT premise
505:     DO setprem
506:     SHOW GETS
507:     mtopic = ALIAS()
508:     SET TOPIC TO &mtopic
509: <═══RETURN
510:
511:
512:     FUNCTION setjoin
513:     PARAMETERS joinop
514:    ┌IF mp > 0
515:    │   K = VAL(premo[mp])
516:    │  ┌IF K > 0
517:    │  │   triple[k,1] = joinop
518:    │  │  ┌IF triple[k,5] > 0
519:    │  │  │   GOTO triple[k,5]
520:    │  │  │   REPLACE premise.op WITH joinop
521:    │  │  └ENDIF
522:    │  └ENDIF
523:    └ENDIF
524:     DO setprem
525:     SHOW GETS
526: <═══RETURN joinop
527:
528:     FUNCTION lastfact
529:     SELECT premise
530:    ┌IF EOF()
531:    │   m.fact = 0
532:    ├ELSE
533:    │   m.fact = fact
534:    └ENDIF
535:     SELECT fact
536: <═══RETURN m.fact
537:
538:     FUNCTION addfact
539:     PRIVATE m.sel, m.flag
540:     m.sel = SELECT()
541:     m.flag = .F.
542:     SELECT fact
543:     GOTO BOTTOM
544:     m.fact = IIF(EOF(), 0, clause) + 1
545:     DO clause.spr WITH m.fact
546:     SEEK m.fact
547:    ┌IF FOUND()
548:    │   SELECT premise
549:    │   APPEND BLANK
550:    │   REPLACE clause WITH m.clause
551:    │   REPLACE fact WITH m.fact
552:    │   m.flag = .T.
553:    └ENDIF
554:     SELECT (m.sel)
555:     DO setprem
556: <═══RETURN m.flag
557:  *:  EOF: TERM.ac1
```

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\KBDELETE.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 07/27/94 at 10:47:58
  8:  *:
  9:  *:         Set by: _QSF0KOKEU()       (function in KBDEL.SPR)
 10:  *:               : _QSF0KOKPT()       (function in KBDEL.SPR)
 11:  *:
 12:  *:           Uses: FACT.DBF
 13:  *:               : PREMISE.DBF
 14:  *:               : ACTION.DBF
 15:  *:               : RULE.DBF
 16:  *:
 17:  *:        Indexes: FACT.IDX
 18:  *:               : PREMISE.IDX
 19:  *:               : ACTION.IDX
 20:  *:               : RULEAREA.IDX
 21:  *:               : SALIENCE.IDX
 22:  *:               : RULE.IDX
 23:  *:
 24:  *:      Documented 15:01:04                                  FoxDoc versio
=> n 3.00a
 25:  *:*************************************************************************
=> *******
 26:      * kbdelete.prg
 27:
 28:      PARAMETERS m.name                          && name of knowledge base
 29:
 30:      * Removes rules and their associated premise, action clauses
 31:      * from the knowledge base.
 32:
 33:      * Lookup the name in the knowledge base file
 34:      SELECT area                                &&
 35:      LOCATE FOR LOWER(name) = LOWER(m.name)     && lookup by name (case inse
=> nsitive)
 36:    ┌─IF !FOUND()                               &&
 37:    │   * unknown knowledge base                &&
 38: <──┘   RETURN                                  &&
 39:    └─ENDIF                                     &&
 40:
 41:      * open database
 42:      SELECT 0
 43:      USE fact INDEX fact
 44:      SELECT 0
 45:      USE premise INDEX premise
 46:      SET RELATION TO fact INTO fact
 47:      SELECT 0
 48:      USE action INDEX action
 49:      SELECT 0
 50:      USE rule INDEX rulearea,salience,rule
 51:      SET RELATION TO premise INTO premise, action INTO action
 52:      SELECT area
 53:      SET RELATION TO area INTO rule
 54:
 55:      * delete rules and clauses in knowledge base
 56:      SELECT rule                                &&
 57:      SEEK area.area
 58:    ┌─DO WHILE area = area.area .AND. !EOF()
 59:    │   SELECT premise
 60:    │   SEEK rule.premise
 61:    │ ┌─DO WHILE clause = rule.premise
 62:    │ │ ┌─IF EMPTY(premise.op)
 63:    │ │ │   SELECT fact
 64:    │ │ │   DELETE FOR clause = premise.fact   && delete all premise cla
=> uses
 65:    │ │ └─ENDIF
 66:    │ │   SELECT premise
 67:    │ │   DELETE
 68:    │ │   SKIP
 69:    │ └─ENDDO
 70:    │   SELECT action
 71:    │   DELETE FOR clause = rule.action       && delete all action clauses
 72:    │   DELETE FOR clause = rule.else         && delete all else clauses
 73:    │   SELECT rule
 74:    │   DELETE                                && delete the rule
 75:    │   SKIP
 76:    └─ENDDO
 77:
 78:      * delete goals and qualifiers in knowledge base
 79:      SELECT goals
 80:      DELETE FOR area = area.area
 81:      PACK
 82:      SELECT DISPLAY
 83:      DELETE FOR area = area.area
 84:      PACK
 85:      SELECT quals
 86:      DELETE FOR area = area.area
 87:      PACK
 88:
 89:      * update premise database
 90:      SELECT premise
 91:      PACK                                      && purge premises
 92:
 93:      * update fact database
 94:      SELECT fact
 95:      PACK                                      && purge facts
 96:
 97:      * update action database
 98:      SELECT action
 99:      PACK                                      && purge actions
100:
101:      * update rulebase
102:      SELECT rule                               &&
103:      PACK                                      && purge rules
104:
105:      * update area database
106:      SELECT area
107:      *replace rules with 0                      && indicate no rules
108:      PRIVATE m.recno
109:      m.recno = RECNO()
110:      DELETE
111:      PACK
112:    ┌─IF m.recno > RECCOUNT()
113:    │   GOTO BOTTOM
114:    ├─ELSE
115:    │   GOTO m.recno
116:    └─ENDIF
117:
118:      *close unuse database
119:      SELECT fact
120:      USE
121:      SELECT premise
122:      USE
123:      SELECT action
124:      USE
125:      SELECT rule
126:      USE
```

```
127:
128:     * finish up
129: <---RETURN
130:
131:  *:  EOF: KBDELETE.act
```

```
*
*
*       08/09/94          OBJECT.SPR                09:39:38
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*
*

PARAMETERS m.status
DO CASE
CASE _WINDOWS

    *       ┌──────────────────────────────────────────────────┐
    *       │                                                  │
    *       │          OBJECT/Windows Setup Code - SECTION 1   │
    *       │                                                  │
    *       └──────────────────────────────────────────────────┘
    *

    #REGION 1
    IF PARAMETER() = 0
       m.status = "Q"
    ENDIF
    m.sel = SELECT()
    SELECT dict
    GOTO BOTTOM
    m.id = id + 1
    m.get = .F.
    DIMENSION marray[1]

    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
       SET TALK OFF
       m.talkstat = "ON"
    ELSE
       m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    m.rborder = SET("READBORDER")
    SET readborder ON

    *       ┌──────────────────────────────────────────────────┐
    *       │                                                  │
    *       │              Windows Window definitions          │
    *       │                                                  │
    *       └──────────────────────────────────────────────────┘
    *

    *       ┌──────────────────────────────────────────────────┐
    *       │                                                  │
    *       │              OBJECT/Windows Screen Layout        │
    *       │                                                  │
    *       └──────────────────────────────────────────────────┘
    *

    #REGION 1
    IF WVISIBLE("object")
       ACTIVATE WINDOW OBJECT SAME
    ELSE
       ACTIVATE WINDOW OBJECT NOSHOW
    ENDIF
    @ 7.385,3.000 SAY "ID# :"  ;
       FONT "MS Sans Serif", 8 ;
       STYLE "B"
    @ 1.462,10.333 GET m.object ;
       PICTURE "@*RHN Subject     ;Disease" ;
       SIZE 1.308,15.000,0.000 ;
       DEFAULT 1 ;
       FONT "MS Sans Serif", 8 ;
       STYLE "B" ;
       VALID _qsf0kpfr1()
    @ 4.385,12.333 GET m.name ;
       SIZE 1.154,38.000 ;
       DEFAULT " " ;
       FONT "MS Sans Serif", 8 ;
       VALID _qsf0kpfv8()
    @ 7.385,10.500 GET m.id ;
       SIZE 1.154,5.000 ;
       DEFAULT 0 ;
       FONT "MS Sans Serif", 8 ;
       DISABLE

IF NOT WEXIST("object") ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.PJX" ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.SCX" ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.MNX" ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.PRG" ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.FRX" ;
      OR UPPER(WTITLE("OBJECT")) == "OBJECT.QPR"
   DEFINE WINDOW OBJECT ;
      AT  0.000, 0.000  ;
      SIZE 12.769,54.333 ;
      TITLE "Add New Object" ;
      FONT "MS Sans Serif", 8 ;
      STYLE "B" ;
      NOFLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      COLOR RGB(,,,128,128,0)
   MOVE WINDOW OBJECT CENTER
ENDIF
```

```
118:     @ 9.692,13.167 GET m.button ;
119:        PICTURE "@*HT \<Ok;\<Cancel" ;
120:        SIZE 1.692,9.833,1.000 ;
121:        DEFAULT 1 ;
122:        FONT "MS Sans Serif", 8 ;
123:        STYLE "B" ;
124:        VALID _qsf0kpfyy()
125:     @ 1.615,3.000 SAY "Type:"  ;
126:        FONT "MS Sans Serif", 8 ;
127:        STYLE "B"
128:     @ 4.231,3.000 GET m.obj ;
129:        PICTURE "@*HN Object" ;
130:        SIZE 1.769,8.000,1.000 ;
131:        DEFAULT 1 ;
132:        FONT "MS Sans Serif", 8 ;
133:        STYLE "B" ;
134:        VALID _qsf0kpg47()
135:
136:   IF NOT WVISIBLE("object")
137:      ACTIVATE WINDOW OBJECT
138:   ENDIF
139:
140:     READ CYCLE MODAL
141:
142:     RELEASE WINDOW OBJECT
143:
144:     #REGION 0
145:
146:     SET readborder &rborder
147:
148:   IF m.talkstat = "ON"
149:      SET TALK ON
150:   ENDIF
151:   IF m.compstat = "ON"
152:      SET COMPATIBLE ON
153:   ENDIF
154:
155:
156: CASE _DOS
157:
158:
159:     *        ┌──────────────────────────────────────────────────────┐
160:     *        │                                                      │
161:     *        │          OBJECT/MS-DOS Setup Code - SECTION 1        │
162:     *        │                                                      │
163:     *        └──────────────────────────────────────────────────────┘
164:     *
165:
166:     #REGION 1
167:   IF PARAMETER() = 0
168:      m.status = "Q"
169:   ENDIF
170:     m.sel = SELECT()
171:     SELECT dict
172:     GOTO BOTTOM
173:     m.id = id + 1
174:     m.get = .F.
175:     DIMENSION marray[1]
176:
177:     #REGION 0
178:     REGIONAL m.currarea, m.talkstat, m.compstat
179:
180:   IF SET("TALK") = "ON"
181:      SET TALK OFF
182:      m.talkstat = "ON"
183:   ELSE
184:      m.talkstat = "OFF"
185:   ENDIF
186:     m.compstat = SET("COMPATIBLE")
187:     SET COMPATIBLE FOXPLUS
188:
189:     *        ┌──────────────────────────────────────────────────────┐
190:     *        │                                                      │
191:     *        │              MS-DOS Window definitions               │
192:     *        │                                                      │
193:     *        └──────────────────────────────────────────────────────┘
194:     *
195:
196:   IF NOT WEXIST("object") ;
197:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.PJX" ;
198:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.SCX" ;
199:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.MNX" ;
200:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.PRG" ;
201:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.FRX" ;
202:         OR UPPER(WTITLE("OBJECT")) == "OBJECT.QPR"
203:      DEFINE WINDOW OBJECT ;
204:         FROM INT((SROW()-11)/2),INT((SCOL()-54)/2) ;
205:         TO INT((SROW()-11)/2)+10,INT((SCOL()-54)/2)+53 ;
206:         TITLE "Add New Object" ;
207:         NOFLOAT ;
208:         NOCLOSE ;
209:         SHADOW ;
210:         NOMINIMIZE ;
211:         COLOR SCHEME 1
212:   ENDIF
213:
214:
215:     *        ┌──────────────────────────────────────────────────────┐
216:     *        │                                                      │
217:     *        │              OBJECT/MS-DOS Screen Layout             │
218:     *        │                                                      │
219:     *        └──────────────────────────────────────────────────────┘
220:     *
221:
222:     #REGION 1
223:   IF WVISIBLE("object")
224:      ACTIVATE WINDOW OBJECT SAME
225:   ELSE
226:      ACTIVATE WINDOW OBJECT NOSHOW
227:   ENDIF
228:     @ 6,1 SAY "ID# :" ;
229:        SIZE 1,5, 0
230:     @ 2,8 GET m.object ;
231:        PICTURE "@*RHN Subject     ;Disease" ;
232:        SIZE 1,16,0 ;
233:        DEFAULT 1 ;
234:        VALID _qsf0kpgoj()
```

```
235:        @ 4,10 GET m.name ;
236:           SIZE 1,38 ;
237:           DEFAULT " " ;
238:           VALID _qsf0kpgsr()
239:        @ 6,8 GET m.id ;
240:           SIZE 1,5 ;
241:           DEFAULT 0 ;
242:           DISABLE
243:        @ 8,19 GET m.button ;
244:           PICTURE "@*HT \<Ok;\<Cancel" ;
245:           SIZE 1,8,1 ;
246:           DEFAULT 1 ;
247:           VALID _qsf0kpgvz()
248:        @ 2,1 SAY "Type:" ;
249:           SIZE 1,5, 0
250:        @ 4,1 GET m.obj ;
251:           PICTURE "@*HN Object" ;
252:           SIZE 1,8,1 ;
253:           DEFAULT 1 ;
254:           VALID _qsf0kph17()
255:
256:     IF NOT WVISIBLE("object")
257:        ACTIVATE WINDOW OBJECT
258:     ENDIF
259:
260:     READ CYCLE MODAL
261:
262:     RELEASE WINDOW OBJECT
263:
264:     #REGION 0
265:     IF m.talkstat = "ON"
266:        SET TALK ON
267:     ENDIF
268:     IF m.compstat = "ON"
269:        SET COMPATIBLE ON
270:     ENDIF
271:
272:
273:  ENDCASE
274:
275:
276:  *
277:  *
278:  *        _QSF0KPFR1            m.object VALID
279:  *
280:  *        Function Origin:
281:  *
282:  *        From Platform:        Windows
283:  *        From Screen:          OBJECT,     Record Number:    3
284:  *        Variable:             m.object
285:  *        Called By:            VALID Clause
286:  *        Object Type:          Radio Button
287:  *        Snippet Number:       1
288:  *
289:  *
290:  *
291:  FUNCTION _qsf0kpfr1     &&  m.object VALID
292:  #REGION 1
293:  IF m.object = 1
294:     SELECT dict
295:     SET ORDER TO 1
296:     GOTO BOTTOM
297:     m.id = id + 1
298:  ELSE
299:     IF m.object = 2
300:        SELECT disease
```



```
301:        SET ORDER TO 1
302:        GOTO BOTTOM
303:        m.id = id + 1
304:     ENDIF
305:  ENDIF
306:  SHOW GETS
307:
308:  *
309:  *
310:  *        _QSF0KPFV8            m.name VALID
311:  *
312:  *        Function Origin:
313:  *
314:  *        From Platform:        Windows
315:  *        From Screen:          OBJECT,     Record Number:    4
316:  *        Variable:             m.name
317:  *        Called By:            VALID Clause
318:  *        Object Type:          Field
319:  *        Snippet Number:       2
320:  *
321:  *
322:  *
323:  FUNCTION _qsf0kpfv8     &&  m.name VALID
324:  #REGION 1
325:  =getobj(m.name)
326:
327:  *
328:  *
329:  *        _QSF0KPFYY            m.button VALID
330:  *
331:  *        Function Origin:
332:  *
333:  *        From Platform:        Windows
334:  *        From Screen:          OBJECT,     Record Number:    6
335:  *        Variable:             m.button
336:  *        Called By:            VALID Clause
337:  *        Object Type:          Push Button
338:  *        Snippet Number:       3
339:  *
340:  *
341:  *
342:  FUNCTION _qsf0kpfyy     &&  m.button VALID
343:  #REGION 1
344:  IF m.button = 1
345:     * SEEK m.id
346:     * IF !found()
347:     APPEND BLANK
348:     REPLACE id WITH m.id
349:     REPLACE name WITH m.name
350:     *          IF !m.get
351:     *             = datainput()
352:     *          ENDIF
353:     * ENDIF
354:     DO CASE
355:     CASE m.status = "Q"
356:        SELECT quals
357:     CASE m.status = "G"
358:        SELECT goals
359:     ENDCASE
360:     APPEND BLANK
361:     REPLACE id WITH m.id
362:     REPLACE OBJECT WITH IIF(m.object=1,"S","D")
363:     REPLACE area WITH area.area
364:     SELECT (m.sel)
365:  ENDIF
366:
```

```
367:    *
368:    *
369:    *          _QSF0KPG47            m.obj VALID
370:    *
371:    *          Function Origin:
372:    *
373:    *          From Platform:        Windows
374:    *          From Screen:          OBJECT,     Record Number:    8
375:    *          Variable:             m.obj
376:    *          Called By:            VALID Clause
377:    *          Object Type:          Push Button
378:    *          Snippet Number:       4
379:    *
380:    *
381:    *
382:    FUNCTION _qsf0kpg47     &&  m.obj VALID
383:    #REGION 1
384:    DO CASE
385:    CASE m.object = 1
386:       SELECT dict
387:    CASE m.object = 2
388:       SELECT disease
389:    ENDCASE
390:    =creatarray()
391:    DO selitem
392:    SHOW GETS
393:
394:
395:    *
396:    *
397:    *          _QSF0KPGOJ            m.object VALID
398:    *
399:    *          Function Origin:
400:    *
401:    *          From Platform:        MS-DOS
402:    *          From Screen:          OBJECT,     Record Number:    13
403:    *          Variable:             m.object
404:    *          Called By:            VALID Clause
405:    *          Object Type:          Radio Button
406:    *          Snippet Number:       5
407:    *
408:    *
409:    *
410:    FUNCTION _qsf0kpgoj     &&  m.object VALID
411:    #REGION 1
412:    IF m.object = 1
413:       SELECT dict
414:       SET ORDER TO 1
415:       GOTO BOTTOM
416:       m.id = id + 1
417:    ELSE
418:       IF m.object = 2
419:          SELECT disease
420:          SET ORDER TO 1
421:          GOTO BOTTOM
422:          m.id = id + 1
423:       ENDIF
424:    ENDIF
425:    SHOW GETS
426:
427:    *
428:    *
429:    *          _QSF0KPGSR            m.name VALID
430:    *
431:    *          Function Origin:
432:    *
433:    *          From Platform:        MS-DOS
434:    *          From Screen:          OBJECT,     Record Number:    14
435:    *          Variable:             m.name
436:    *          Called By:            VALID Clause
437:    *          Object Type:          Field
438:    *          Snippet Number:       6
439:    *
440:    *
441:    *
442:    FUNCTION _qsf0kpgsr     &&  m.name VALID
443:    #REGION 1
444:    =getobj(m.name)
445:
446:    *
447:    *
448:    *          _QSF0KPGVZ            m.button VALID
449:    *
450:    *          Function Origin:
451:    *
452:    *          From Platform:        MS-DOS
453:    *          From Screen:          OBJECT,     Record Number:    16
454:    *          Variable:             m.button
455:    *          Called By:            VALID Clause
456:    *          Object Type:          Push Button
457:    *          Snippet Number:       7
458:    *
459:    *
460:    *
461:    FUNCTION _qsf0kpgvz     &&  m.button VALID
462:    #REGION 1
463:    IF m.button = 1
464:       * SEEK m.id
465:       * IF !found()
466:       APPEND BLANK
467:       REPLACE id WITH m.id
468:       REPLACE name WITH m.name
469:       *          IF !m.get
470:       *             = datainput()
471:       *          ENDIF
472:       * ENDIF
473:       DO CASE
474:       CASE m.status = "Q"
475:          SELECT quals
476:       CASE m.status = "G"
477:          SELECT goals
478:       ENDCASE
479:       APPEND BLANK
480:       REPLACE id WITH m.id
481:       REPLACE OBJECT WITH IIF(m.object=1,"S","D")
482:       REPLACE area WITH area.area
483:       SELECT (m.sel)
484:    ENDIF
485:
486:    *
487:    *
488:    *          _QSF0KPH17            m.obj VALID
489:    *
490:    *          Function Origin:
491:    *
492:    *          From Platform:        MS-DOS
493:    *          From Screen:          OBJECT,     Record Number:    18
494:    *          Variable:             m.obj
495:    *          Called By:            VALID Clause
496:    *          Object Type:          Push Button
497:    *          Snippet Number:       8
498:    *
```

```
499:     *
500:     *
501:     FUNCTION _qsf0kph17     &&  m.obj VALID
502:     #REGION 1
503:    ┌DO CASE
504:    ├CASE m.object = 1
505:    │   SELECT dict
506:    ├CASE m.object = 2
507:    │   SELECT disease
508:    └ENDCASE
509:     =creatarray()
510:     DO selitem
511:     SHOW GETS
512:
513:
514:
515:     *
516:     *
517:     *          OBJECT/MS-DOS Supporting Procedures and Functions
518:     *
519:     *
520:     *
521:
522:     #REGION 1
523:     FUNCTION creatarray
524:    ┌IF RECCOUNT() > 0
525:    │   DIMENSION marray[reccount()]
526:    ├ELSE
527:    │   DIMENSION marray[1]
528:    └ENDIF
529:     marray = ""
530:     i = 0
531:    ┌IF m.status = "Q"
532:    │   m.dbf = 'QUALS'
533:    ├ELSE
534:    │   m.dbf = 'GOALS'
535:    └ENDIF
536:     GOTO TOP
537:    ┌SCAN
538:    │   ┌IF EMPTY(SEEK(id,m.dbf))
539:    │   │   i = i + 1
540:    │   │   marray[i] = RIGHT(STR(id),5) + "  " + TRIM(name)
541:    │   └ENDIF
542:    └ENDSCAN
543:    ┌IF i > 0
544:    │   DECLARE marray[i] && resize marray
545:    ├ELSE
546:    │   DECLARE marray[1]
547:    │   marray = ""
548:    └ENDIF
549: <──RETURN
550:
551:     FUNCTION datainput
552:     m.get = .T.
553:    ┌DO CASE
554:    ├CASE "DISEASE.DBF" $ DBF()
555:    │   DO disease.spr
556:    ├CASE "DICT.DBF" $ DBF()
557:    │   DO dict.spr WITH m.id
558:    └ENDCASE
559:     m.id = id
560:     m.name = name
561: <──RETURN
562:
563:     *--------------------------------------------
564:     *
```

```
565:     *  Get the matching list of object
566:     *
567:     *--------------------------------------------
568:
569:     FUNCTION getobj
570:     PARAMETER m.name
571:     PRIVATE m.file, mcom, msel,m.temp
572:     mcom = SET("EXACT")
573:     SET EXACT OFF
574:     m.file = DBF()
575:     msel = SELECT()
576:     m.temp = m.name + "%"
577:
578:     SELECT id, name;
579:        FROM (m.file);
580:        INTO CURSOR t0000000;
581:        WHERE UPPER(name) LIKE (UPPER(m.temp))
582:     SELECT t0000000
583:     =creatarray()
584:     USE
585:     SELECT (msel)
586:     DO selitem
587:     *delete file t0000000.dbf
588:     SET EXACT &mcom
589: <──RETURN
590:
591:
592:     FUNCTION selitem
593:     m.item = ""
594:    ┌IF !EMPTY(marray[1])
595:    │   DO oblist.spr WITH marray, m.item
596:    │   ┌IF m.item $ "~"
597:    │   │   m.name = ""
598:    │   │   _CUROBJ = OBJNUM(m.name)
599:    │   │   SHOW GETS
600: <──┼───┼──RETURN
601:    │   └ENDIF
602:    └ENDIF
603:    ┌IF EMPTY(m.item)
604:    │   m.get = .T.
605:    │   ┌DO CASE
606:    │   ├CASE "DISEASE.DBF" $ DBF()
607:    │   │   DO disease.spr
608:    │   ├CASE "DICT.DBF" $ DBF()
609:    │   │   DO dict.spr WITH m.id
610:    │   └ENDCASE
611:    │   m.id = id
612:    │   m.name = name
613:    ├ELSE
614:    │   m.id = VAL(m.item)
615:    │   m.name = TRIM(SUBSTR(m.item,7,LEN(m.item)))
616:    └ENDIF
617: *:  EOF: OBJECT.ac1
```

```
*
*
*      08/09/94            ENUM.SPR            09:39:43
*
*
*
*      Author's Name
*
*      Copyright (c) 1994 Company Name
*      Address
*      City,     Zip
*
*      Description:
*      This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *
   *
   *              Windows Window definitions
   *
   *
   *

   IF NOT WEXIST("w_qenum") ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.PJX" ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.SCX" ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.MNX" ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.PRG" ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.FRX" ;
         OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.QPR"
      DEFINE WINDOW w_qenum ;
         AT  0.000, 0.000  ;
         SIZE 10.250,58.125 ;
         TITLE "Enumerated Type Editor" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE
      MOVE WINDOW w_qenum CENTER
   ENDIF


   *
   *
   *              ENUM/Windows Screen Layout
   *
   *
   *

   #REGION 1
   IF WVISIBLE("w_qenum")
      ACTIVATE WINDOW w_qenum SAME
   ELSE
      ACTIVATE WINDOW w_qenum NOSHOW
   ENDIF
   @ 7.333,24.000 GET mbuttons ;
      PICTURE "@*HT OK;Cancel" ;
      SIZE 1.917,7.500,0.750 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kpjam()
   @ 4.583,1.625 SAY "Ord"  ;
      FONT "Terminal", 8
   @ 0.583,1.500 SAY "Mutex"  ;
      FONT "Terminal", 8
   @ 4.583,7.250 GET enum.ord ;
      SIZE 1.000,2.000 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      DISABLE
   @ 0.750,7.500 GET enum.mutex ;
      SIZE 1.000,1.000 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      VALID _qsf0kpjf7()
   @ 2.500,7.500 GET enum.enumerate ;
      SIZE 1.333,48.125 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      PICTURE "@!"
   @ 2.583,1.500 SAY "Enum"  ;
      FONT "Terminal", 8
   @ 7.250,17.375 GET mbbutton ;
      PICTURE "@*VN Add" ;
      SIZE 1.917,6.000,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8 ;
      VALID _qsf0kpjjk()

   IF NOT WVISIBLE("w_qenum")
      ACTIVATE WINDOW w_qenum
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_qenum

   #REGION 0

   SET readborder &rborder
```

```
    IF m.talkstat = "ON"
       SET TALK ON
    ENDIF
    IF m.compstat = "ON"
       SET COMPATIBLE ON
    ENDIF


CASE _DOS


    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
       SET TALK OFF
       m.talkstat = "ON"
    ELSE
       m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    *     ┌──────────────────────────────────────────────────┐
    *     │                                                  │
    *     │                MS-DOS Window definitions         │
    *     │                                                  │
    *     └──────────────────────────────────────────────────┘
    *

    IF NOT WEXIST("w_qenum") ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.PJX" ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.SCX" ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.MNX" ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.PRG" ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.FRX" ;
          OR UPPER(WTITLE("W_QENUM")) == "W_QENUM.QPR"
       DEFINE WINDOW w_qenum ;
          FROM INT((SROW()-6)/2),INT((SCOL()-71)/2) ;
          TO INT((SROW()-6)/2)+5,INT((SCOL()-71)/2)+70 ;
          TITLE "Enumerated Type Editor" ;
          FLOAT ;
          CLOSE ;
          SHADOW ;
          NOMINIMIZE ;
          COLOR SCHEME 1
    ENDIF


    *     ┌──────────────────────────────────────────────────┐
    *     │                                                  │
    *     │                ENUM/MS-DOS Screen Layout         │
    *     │                                                  │
    *     └──────────────────────────────────────────────────┘
    *
```

```
    #REGION 1
    IF WVISIBLE("w_qenum")
       ACTIVATE WINDOW w_qenum SAME
    ELSE
       ACTIVATE WINDOW w_qenum NOSHOW
    ENDIF
    @ 3,32 GET mbuttons ;
       PICTURE "@*HT OK;Cancel" ;
       SIZE 1,8,1 ;
       DEFAULT 1 ;
       VALID _qsf0kpjyy()
    @ 2,0 SAY "Ord" ;
       SIZE 1,3, 0
    @ 0,0 SAY "Mutex" ;
       SIZE 1,5, 0
    @ 2,6 GET enum.ord ;
       SIZE 1,2 ;
       DEFAULT " " ;
       DISABLE
    @ 0,6 GET enum.mutex ;
       SIZE 1,1 ;
       DEFAULT " " ;
       VALID _qsf0kpk3s()
    @ 1,6 GET enum.enumerate ;
       SIZE 1,63 ;
       DEFAULT " " ;
       PICTURE "@!"
    @ 1,0 SAY "Enum" ;
       SIZE 1,4, 0
    @ 3,22 GET mbbutton ;
       PICTURE "@*VN Add" ;
       SIZE 1,8,1 ;
       DEFAULT 1 ;
       VALID _qsf0kpk8e()

    IF NOT WVISIBLE("w_qenum")
       ACTIVATE WINDOW w_qenum
    ENDIF

    READ CYCLE MODAL

    RELEASE WINDOW w_qenum

    #REGION 0
    IF m.talkstat = "ON"
       SET TALK ON
    ENDIF
    IF m.compstat = "ON"
       SET COMPATIBLE ON
    ENDIF


ENDCASE


*
*     _QSF0KPJAM          mbuttons VALID
*
*     Function Origin:
*
*     From Platform:      Windows
*     From Screen:        ENUM,     Record Number:    2
*     Variable:           mbuttons
*     Called By:          VALID Clause
*     Object Type:        Push Button
```

```
245:     *         Snippet Number:      1
246:     *
247:     *
248:     *
249:     FUNCTION _qsf0kpjam     &&  mbuttons VALID
250:     #REGION 1
251:    ┌DO CASE
252:    ├CASE mbuttons = 1   && ok
253:    │   mok = .T.
254:    ├CASE mbuttons = 2   && cancel
255:    │   mok = .F.
256:    └ENDCASE
257:
258:     *
259:     *
260:     *         _QSF0KPJF7            enum.mutex VALID
261:     *
262:     *         Function Origin:
263:     *
264:     *         From Platform:        Windows
265:     *         From Screen:          ENUM,     Record Number:    6
266:     *         Variable:             enum.mutex
267:     *         Called By:            VALID Clause
268:     *         Object Type:          Field
269:     *         Snippet Number:       2
270:     *
271:     *
272:     *
273:     FUNCTION _qsf0kpjf7     &&  enum.mutex VALID
274:     #REGION 1
275:    ┌DO CASE
276:    ├CASE dict.datatype $ "EL"
277: <──────RETURN mutex = "-"
278:    ├CASE dict.datatype $ "M"
279: <──────RETURN mutex $ "-+"
280:    └ENDCASE
281: <──RETURN .F.
282:
283:     *
284:     *
285:     *         _QSF0KPJJK            mbbutton VALID
286:     *
287:     *         Function Origin:
288:     *
289:     *         From Platform:        Windows
290:     *         From Screen:          ENUM,     Record Number:    9
291:     *         Variable:             mbbutton
292:     *         Called By:            VALID Clause
293:     *         Object Type:          Push Button
294:     *         Snippet Number:       3
295:     *
296:     *
297:     *
298:     FUNCTION _qsf0kpjjk     &&  mbbutton VALID
299:     #REGION 1
300:     m.ord = 0
301:    ┌DO WHILE id = dict.id
302:    │   m.ord = ord
303:    │   SKIP
304:    └ENDDO
305:     m.ord  =  m.ord + 1
306:     APPEND BLANK
307:     m.id  =  dict.id
308:     m.mutex  =  "-"
309:     GATHER MEMVAR
310:     _CUROBJ = OBJNUM(enum.mutex)
```



```
311:     SHOW GETS
312:
313:
314:     *
315:     *
316:     *         _QSF0KPJYY            mbuttons VALID
317:     *
318:     *         Function Origin:
319:     *
320:     *         From Platform:        MS-DOS
321:     *         From Screen:          ENUM,     Record Number:   12
322:     *         Variable:             mbuttons
323:     *         Called By:            VALID Clause
324:     *         Object Type:          Push Button
325:     *         Snippet Number:       4
326:     *
327:     *
328:     *
329:     FUNCTION _qsf0kpjyy     &&  mbuttons VALID
330:     #REGION 1
331:    ┌DO CASE
332:    ├CASE mbuttons = 1   && ok
333:    │   mok = .T.
334:    ├CASE mbuttons = 2   && cancel
335:    │   mok = .F.
336:    └ENDCASE
337:
338:     *
339:     *
340:     *         _QSF0KPK3S            enum.mutex VALID
341:     *
342:     *         Function Origin:
343:     *
344:     *         From Platform:        MS-DOS
345:     *         From Screen:          ENUM,     Record Number:   16
346:     *         Variable:             enum.mutex
347:     *         Called By:            VALID Clause
348:     *         Object Type:          Field
349:     *         Snippet Number:       5
350:     *
351:     *
352:     *
353:     FUNCTION _qsf0kpk3s     &&  enum.mutex VALID
354:     #REGION 1
355:    ┌DO CASE
356:    ├CASE dict.datatype $ "EL"
357: <──────RETURN mutex = "-"
358:    ├CASE dict.datatype $ "M"
359: <──────RETURN mutex $ "-+"
360:    └ENDCASE
361: <──RETURN .F.
362:
363:     *
364:     *
365:     *         _QSF0KPK8E            mbbutton VALID
366:     *
367:     *         Function Origin:
368:     *
369:     *         From Platform:        MS-DOS
370:     *         From Screen:          ENUM,     Record Number:   19
371:     *         Variable:             mbbutton
372:     *         Called By:            VALID Clause
373:     *         Object Type:          Push Button
374:     *         Snippet Number:       6
375:     *
376:     *
```

```
377:     *
378:     FUNCTION _qsf0kpk8e     &&  mbbutton VALID
379:     #REGION 1
380:     m.ord = 0
381:     DO WHILE id = dict.id
382:        m.ord = ord
383:        SKIP
384:     ENDDO
385:     m.ord  =  m.ord + 1
386:     APPEND BLANK
387:     m.id  =  dict.id
388:     m.mutex  =  "-"
389:     GATHER MEMVAR
390:      CUROBJ = OBJNUM(enum.mutex)
391:     SHOW GETS
392:  *:  EOF: ENUM.ac1
```

```
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *

PARAMETERS MESSAGE, wildcard, filename
DO CASE
CASE _WINDOWS

    *
    *
    *
    *
    *
    *
    *

#REGION 1
PRIVATE mfanme, mpath, olddrive, predrive, oldpath, maction, mdriv
=> e, mfiles
    DO CASE
    CASE PARAMETERS() = 0
        m.message = ""
        m.wildcard = "*.zip"
        mfname = ""
    CASE PARAMETERS() = 1 OR EMPTY(m.wildcard)
        m.wildcard = "*.zip"
        mfname = ""
    CASE PARAMETERS() = 2 OR EMPTY(m.filename)
        mfname = ""
    OTHERWISE
        mfname = filename
    ENDCASE

    #REGION 0
    REGIONAL m.currarea, m.talkstat, m.compstat

    IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
    ELSE
        m.talkstat = "OFF"
    ENDIF
    m.compstat = SET("COMPATIBLE")
    SET COMPATIBLE FOXPLUS

    m.rborder = SET("READBORDER")
    SET readborder ON
```

```
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*

IF NOT WEXIST("w_restore") ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.PJX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.SCX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.MNX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.PRG" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.FRX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.QPR"
   DEFINE WINDOW w_restore ;
      AT  0.000, 0.000  ;
      SIZE 18.846,79.500 ;
      FONT "MS Sans Serif", 8 ;
      STYLE "B" ;
      NOFLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      DOUBLE
   MOVE WINDOW w_restore CENTER
ENDIF

    *
    *
    *
    *
    *
    *
    *

#REGION 1
m.olddrive = SET("DEFAULT")
m.prevdrive = m.olddrive
m.oldpath = CURDIR()
maction = 1

DECLARE drivearray[1,1]
mdrive = 1
DO newdrivepop

DECLARE patharray[1,1]
mpath = 1
DO newpathpop

DECLARE filearray[1,1]
mfiles = 1
DO newfilepop


    *
```

```
*
*                    RESTORE/Windows Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("w_restore")
   ACTIVATE WINDOW w_restore SAME
ELSE
   ACTIVATE WINDOW w_restore NOSHOW
ENDIF
@ 3.692,59.500 GET mdrive ;
   PICTURE "@^" ;
   FROM drivearray ;
   SIZE 1.500,12.500 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   WHEN _qsf0kpmpz() ;
   VALID _qsf0kpms0()
@ 7.154,42.667 SAY "Directory:"  ;
   FONT "Terminal", 8
@ 0.615,3.167 SAY MESSAGE ;
   SIZE 1.000,39.875 ;
   FONT "Terminal", 8
@ 7.000,59.500 GET mpath ;
   PICTURE "@^" ;
   FROM patharray ;
   SIZE 1.500,12.500 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpmxi()
@ 10.231,43.333 GET maction ;
   PICTURE "@*HT \!\<Restore;\<Cancel" ;
   SIZE 2.250,11.375,1.625 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpn0j()
@ 3.154,2.833 GET mfile ;
   PICTURE "@&N" ;
   FROM filearray ;
   SIZE 11.667,28.875 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpn3t()
@ 15.846,29.000 GET mfname ;
   SIZE 1.000,32.500 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpn7e()
@ 15.846,3.167 SAY "Restore file name: "  ;
   FONT "Terminal", 8
@ 3.923,42.667 SAY "From drive: "  ;
   FONT "Terminal", 8

IF NOT WVISIBLE("w_restore")
   ACTIVATE WINDOW w_restore
ENDIF

READ CYCLE MODAL
RELEASE WINDOW w_restore

#REGION 0

SET readborder &rborder

IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


*
*                    RESTORE/Windows Cleanup Code
*
*
*

#REGION 1
SET DEFAULT TO (m.olddrive + m.oldpath)
RETURN

*********************************************************

**********************

CASE _DOS

*
*                    RESTORE/MS-DOS Setup Code - SECTION 1
*
*
*

#REGION 1
PRIVATE mfanme, mpath, olddrive, predrive, oldpath, maction, mdrive, mfiles
   DO CASE
   CASE PARAMETERS() = 0
      m.message = ""
      m.wildcard = "*.zip"
      mfname = ""
   CASE PARAMETERS() = 1 OR EMPTY(m.wildcard)
      m.wildcard = "*.zip"
      mfname = ""
   CASE PARAMETERS() = 2 OR EMPTY(m.filename)
      mfname = ""
   OTHERWISE
      mfname = filename
   ENDCASE
```

```
#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

*       ┌──────────────────────────────────────────────────────┐
*       │                                                      │
*       │                MS-DOS Window definitions             │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

IF NOT WEXIST("w_restore") ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.PJX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.SCX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.MNX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.PRG" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.FRX" ;
      OR UPPER(WTITLE("W_RESTORE")) == "W_RESTORE.QPR"
   DEFINE WINDOW w_restore ;
      FROM INT((SROW()-18)/2),INT((SCOL()-61)/2) ;
      TO INT((SROW()-18)/2)+17,INT((SCOL()-61)/2)+60 ;
      NOFLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      DOUBLE ;
      COLOR SCHEME 5
ENDIF

*       ┌──────────────────────────────────────────────────────┐
*       │                                                      │
*       │          RESTORE/MS-DOS Setup Code - SECTION 2       │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

#REGION 1
m.olddrive = SET("DEFAULT")
m.prevdrive = m.olddrive
m.oldpath = CURDIR()
maction = 1

DECLARE drivearray[1,1]
mdrive = 1
DO newdrivepop

DECLARE patharray[1,1]
mpath = 1
DO newpathpop

DECLARE filearray[1,1]
mfiles = 1
DO newfilepop

*       ┌──────────────────────────────────────────────────────┐
*       │                                                      │
*       │              RESTORE/MS-DOS Screen Layout            │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

#REGION 1
IF WVISIBLE("w_restore")
   ACTIVATE WINDOW w_restore SAME
ELSE
   ACTIVATE WINDOW w_restore NOSHOW
ENDIF
@ 2,40 GET mdrive ;
   PICTURE "@^" ;
   FROM drivearray ;
   SIZE 3,18 ;
   DEFAULT 1 ;
   WHEN _qsf0kpny5() ;
   VALID _qsf0kpo0a() ;
   COLOR SCHEME 5, 6
@ 6,28 SAY "Directory:" ;
   SIZE 1,10, 0
@ 0,0 SAY MESSAGE ;
   SIZE 1,40
@ 5,40 GET mpath ;
   PICTURE "@^" ;
   FROM patharray ;
   SIZE 3,18 ;
   DEFAULT 1 ;
   VALID _qsf0kpo5t() ;
   COLOR SCHEME 5, 6
@ 9,45 GET maction ;
   PICTURE "@*VT \!\<Restore;\<Cancel" ;
   SIZE 1,9,1 ;
   DEFAULT 1 ;
   VALID _qsf0kpo8t()
@ 2,1 GET mfile ;
   PICTURE "@&N" ;
   FROM filearray ;
   SIZE 11,26 ;
   DEFAULT 1 ;
   VALID _qsf0kpoby() ;
   COLOR SCHEME 6
@ 14,19 GET mfname ;
   SIZE 1,39 ;
   DEFAULT " " ;
   VALID _qsf0kpofg()
@ 14,0 SAY "Restore file name: " ;
   SIZE 1,19, 0
@ 3,28 SAY "From drive: " ;
   SIZE 1,12, 0
```

```
  IF NOT WVISIBLE("w_restore")
     ACTIVATE WINDOW w_restore
  ENDIF

  READ CYCLE MODAL

  RELEASE WINDOW w_restore

  #REGION 0
  IF m.talkstat = "ON"
     SET TALK ON
  ENDIF
  IF m.compstat = "ON"
     SET COMPATIBLE ON
  ENDIF


  *
  *
  *            RESTORE/MS-DOS Cleanup Code
  *
  *

  *

  #REGION 1
  SET DEFAULT TO (m.olddrive + m.oldpath)
  RETURN

  ******************************************************

  *********************

ENDCASE


*
*
*     _QSFOKPMPZ          mdrive WHEN
*
*     Function Origin:
*
*     From Platform:      Windows
*     From Screen:        RESTORE,     Record Number:    2
*     Variable:           mdrive
*     Called By:          WHEN Clause
*     Object Type:        Popup
*     Snippet Number:     1
*
*
*
FUNCTION _qsf0kpmpz     &&  mdrive WHEN
#REGION 1
m.prevdrive = mdrive

*
*
*     _QSFOKPMS0          mdrive VALID
*
*     Function Origin:
*
*     From Platform:      Windows
*     From Screen:        RESTORE,     Record Number:    2
*     Variable:           mdrive
*     Called By:          VALID Clause
*     Object Type:        Popup
*     Snippet Number:     2
*
*
*
*Switch to the selected drive
FUNCTION _qsf0kpms0     &&  mdrive VALID
#REGION 1
PRIVATE newdrive,mready

*Convert the popup bar number into the matching drive name
m.newdrive = drivearray[mdrive]

IF UPPER(m.newdrive) $ "A:B:"
   mready = yesno("Please insert disk into drive " + m.newdrive, "rea
=> dy","cancel")
ELSE
   mready = .T.
ENDIF
IF mready
   *Go there and reset all the other popups to match
   SET DEFAULT TO (m.newdrive)
   DO newpathpop
   DO newfilepop
ELSE
   mdrive =  m.prevdrive
   m.newdrive = drivearray[mdrive]
ENDIF
SHOW GETS

 *
 *
 *     _QSFOKPMXI          mpath VALID
 *
 *     Function Origin:
 *
 *     From Platform:      Windows
 *     From Screen:        RESTORE,     Record Number:    5
 *     Variable:           mpath
 *     Called By:          VALID Clause
 *     Object Type:        Popup
 *     Snippet Number:     3
 *
 *
 *
 FUNCTION _qsf0kpmxi     &&  mpath VALID
 #REGION 1
 m.newdefault = pathstring()
 SET DEFAULT TO (m,newdefault)
 DO newpathpop
 DO newfilepop
 SHOW GETS

 *
 *
 *     _QSFOKPNOJ          maction VALID
 *
 *     Function Origin:
 *
 *     From Platform:      Windows
 *     From Screen:        RESTORE,     Record Number:    6
 *     Variable:           maction
```

```
476:     *          Called By:          VALID Clause
477:     *          Object Type:        Push Button
478:     *          Snippet Number:     4
479:     *
480:     *
481:     *
482:     FUNCTION _qsf0kpn0j     &&  maction VALID
483:     #REGION 1
484:     DO CASE
485:     CASE maction = 1
486:        IF FILE(mfname)
487:           DO pkunzip
488:        ENDIF
489:     OTHERWISE
490:     ENDCASE
491:
492:     *
493:     *          _QSF0KPN3T          mfile VALID
494:     *
495:     *          Function Origin:
496:     *
497:     *          From Platform:      Windows
498:     *          From Screen:        RESTORE,     Record Number:    7
499:     *          Variable:           mfile
500:     *          Called By:          VALID Clause
501:     *          Object Type:        List
502:     *          Snippet Number:     5
503:     *
504:     *
505:     *
506:     FUNCTION _qsf0kpn3t     &&  mfile VALID
507:     #REGION 1
508:     mnewfile = filearray[mfile,1]
509:     IF "[" $ mnewfile
510:        mnewpath = SUBSTR(mnewfile,2,LEN(mnewfile)-2)
511:        SET DEFA TO (mnewpath)
512:        DO newpathpop
513:        DO newfilepop
514:        SHOW GETS
515:     ELSE
516:        mfname = mnewfile
517:        SHOW GETS
518:     ENDIF
519:
520:     *
521:     *          _QSF0KPN7E          mfname VALID
522:     *
523:     *          Function Origin:
524:     *
525:     *          From Platform:      Windows
526:     *          From Screen:        RESTORE,     Record Number:    8
527:     *          Variable:           mfname
528:     *          Called By:          VALID Clause
529:     *          Object Type:        Field
530:     *          Snippet Number:     6
531:     *
532:     *
533:     *
534:     *
535:     FUNCTION _qsf0kpn7e     &&  mfname VALID
536:     #REGION 1
537:     IF !EMPTY(mfname)
538:        IF !FILE(mfname)
539:           mfname = ""
540:        ENDIF
541:
```

```
542:     ENDIF
543:
544:     *
545:     *
546:     *          _QSF0KPNY5          mdrive WHEN
547:     *
548:     *          Function Origin:
549:     *
550:     *          From Platform:      MS-DOS
551:     *          From Screen:        RESTORE,     Record Number:    14
552:     *          Variable:           mdrive
553:     *          Called By:          WHEN Clause
554:     *          Object Type:        Popup
555:     *          Snippet Number:     7
556:     *
557:     *
558:     *
559:     FUNCTION _qsf0kpny5     &&  mdrive WHEN
560:     #REGION 1
561:     m.prevdrive = mdrive
562:
563:     *
564:     *
565:     *          _QSF0KPO0A          mdrive VALID
566:     *
567:     *          Function Origin:
568:     *
569:     *          From Platform:      MS-DOS
570:     *          From Screen:        RESTORE,     Record Number:    14
571:     *          Variable:           mdrive
572:     *          Called By:          VALID Clause
573:     *          Object Type:        Popup
574:     *          Snippet Number:     8
575:     *
576:     *
577:     *
578:     *Switch to the selected drive
579:     FUNCTION _qsf0kpo0a     &&  mdrive VALID
580:     #REGION 1
581:     PRIVATE newdrive,mready
582:
583:     *Convert the popup bar number into the matching drive name
584:     m.newdrive = drivearray[mdrive]
585:
586:     IF UPPER(m.newdrive) $ "A:B:"
587:        mready = yesno("Please insert disk into drive " + m.newdrive, "rea
=> dy","cancel")
588:     ELSE
589:        mready = .T.
590:     ENDIF
591:     IF mready
592:        *Go there and reset all the other popups to match
593:        SET DEFAULT TO (m.newdrive)
594:        DO newpathpop
595:        DO newfilepop
596:     ELSE
597:        mdrive =  m.prevdrive
598:        m.newdrive = drivearray[mdrive]
599:     ENDIF
600:     SHOW GETS
601:
602:     *
603:     *
604:     *          _QSF0KPO5T          mpath VALID
605:     *
606:     *          Function Origin:
```

```
*
*       From Platform:      MS-DOS
*       From Screen:        RESTORE,      Record Number:   17
*       Variable:           mpath
*       Called By:          VALID Clause
*       Object Type:        Popup
*       Snippet Number:     9
*
*
FUNCTION _qsf0kpo5t     &&  mpath VALID
#REGION 1
m.newdefault = pathstring()
SET DEFAULT TO (m.newdefault)
DO newpathpop
DO newfilepop
SHOW GETS

*
*
*       _QSFOKPO8T          maction VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        RESTORE,      Record Number:   18
*       Variable:           maction
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     10
*
*
*
FUNCTION _qsf0kpo8t     &&  maction VALID
#REGION 1
DO CASE
CASE maction = 1
   IF FILE(mfname)
      DO pkunzip
   ENDIF
OTHERWISE
ENDCASE

*
*
*       _QSFOKPOBY          mfile VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        RESTORE,      Record Number:   19
*       Variable:           mfile
*       Called By:          VALID Clause
*       Object Type:        List
*       Snippet Number:     11
*
*
*
FUNCTION _qsf0kpoby     &&  mfile VALID
#REGION 1
mnewfile = filearray[mfile,1]
IF "[" $ mnewfile
   mnewpath = SUBSTR(mnewfile,2,LEN(mnewfile)-2)
   SET DEFA TO (mnewpath)
   DO newpathpop
   DO newfilepop
   SHOW GETS
ELSE
   mfname = mnewfile
   SHOW GETS
ENDIF

*
*
*       _QSFOKPOFG          mfname VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        RESTORE,      Record Number:   20
*       Variable:           mfname
*       Called By:          VALID Clause
*       Object Type:        Field
*       Snippet Number:     12
*
*
*
FUNCTION _qsf0kpofg     &&  mfname VALID
#REGION 1
IF !EMPTY(mfname)
   IF !FILE(mfname)
      mfname = ""
   ENDIF
ENDIF


*
*
*       RESTORE/MS-DOS Supporting Procedures and Functions
*
*
*

#REGION 1

*
*
*       RESTORE Procedure NEWDRIVEPOP
*
*
*


PROCEDURE newdrivepop
DO CASE
CASE _DOS
   *********************
   * Create a popup with each of the legal drive names
   * The popup will be based on an array of the drive names
   PRIVATE olddefault, olderror, drivename, ERROR

   * We will change drives, so remember where we started
   m.olddefault = SET( "DEFAULT" )

   * To test for legal drives this program SET DEFAULT TO each drive
   * If no error occurs the drive name will be added to an array

   * Create the error trap
   m.olderror = ON( "ERROR" )
   m.error = .F.
   ON ERROR m.error = .T.
```

```
739:        * Create the array of legal drive names
740:        DECLARE drivearray[ 1 ]
741:        drivearray[ 1 ] = ""
742:
743:        * Loop from A to Z
744:        m.drivename = "A"
745:        FOR i = 1 TO 26
746:
747:           * Try to switch to the drive
748:           SET DEFAULT TO (m.drivename)
749:
750:           * If it worked
751:           IF NOT m.error
752:
753:              * Add the name to the last element in the array
754:              drivearray[ ALEN( DriveArray ) ] = m.drivename + ":"
755:
756:              * Add another element at the end of the array
757:              DECLARE drivearray[ ALEN( DriveArray ) + 1 ]
758:
759:           ENDIF
760:
761:           * Change to the next letter in the alphabet
762:           m.drivename = CHR( ASC( m.drivename ) + 1 )
763:
764:           * Reset the error trap
765:           m.error = .F.
766:        NEXT
767:
768:        * If the first element is empty, no drives were found
769:        IF EMPTY( drivearray[ 1 ] )
770:           SHOW GET m.drive disabled
771:        ELSE
772:           * Drives were found so cut off the last empty element
773:           * added to the array in the loop
774:           DECLARE drivearray[ ALEN( DriveArray ) - 1 ]
775:        ENDIF
776:
777:        * Reset the error trap and return to home
778:        ON ERROR &olderror
779:        SET DEFAULT TO (m.olddefault)
780:
781:        * Initialize the popup variable to the element in the
782:        * drive array which contains the current drive
783:        mdrive = ASCAN( drivearray, m.olddefault )
784: <======RETURN
785:
786:        **********
787:     CASE  WINDOWS
788:        **********
789:        * Create a popup with each of the legal drive names
790:        * The popup will be based on an array of the drive names
791:        PRIVATE olddefault, olderror, drivename, ERROR
792:
793:        * We will change drives, so remember where we started
794:        m.olddefault = SET( "DEFAULT" )
795:
796:        * To test for legal drives this program SET DEFAULT TO each drive
797:        * If no error occurs the drive name will be added to an array
798:
799:        * Create the error trap
800:        m.olderror = ON( "ERROR" )
801:        m.error = .F.
802:        ON ERROR m.error = .T.
803:
804:        * Create the array of legal drive names
```

```
805:        DECLARE drivearray[ 3 ]
806:        drivearray[ 1 ] = "A"
807:        drivearray[ 2 ] = "B"
808:        drivearray[ 3 ] = ""
809:
810:        * Loop from A to Z
811:        m.drivename = "C"
812:        FOR i = 3 TO 26
813:
814:           * Try to switch to the drive
815:           SET DEFAULT TO (m.drivename)
816:
817:           * If it worked
818:           IF NOT m.error
819:
820:              * Add the name to the last element in the array
821:              drivearray[ ALEN( DriveArray ) ] = m.drivename + ":"
822:
823:              * Add another element at the end of the array
824:              DECLARE drivearray[ ALEN( DriveArray ) + 1 ]
825:
826:           ENDIF
827:
828:           * Change to the next letter in the alphabet
829:           m.drivename = CHR( ASC( m.drivename ) + 1 )
830:
831:           * Reset the error trap
832:           m.error = .F.
833:        NEXT
834:
835:        * If the first element is empty, no drives were found
836:        IF EMPTY( drivearray[ 1 ] )
837:           SHOW GET m.drive disabled
838:        ELSE
839:           * Drives were found so cut off the last empty element
840:           * added to the array in the loop
841:           DECLARE drivearray[ ALEN( DriveArray ) - 1 ]
842:        ENDIF
843:
844:        * Reset the error trap and return to home
845:        ON ERROR &olderror
846:        SET DEFAULT TO (m.olddefault)
847:
848:        * Initialize the popup variable to the element in the
849:        * drive array which contains the current drive
850:        mdrive = ASCAN( drivearray, m.olddefault )
851: <======RETURN
852:
853:        **********
854:     ENDCASE
855:
856:     *
857:     *
858:     *        RESTORE Procedure NEWPATHPOP
859:     *
860:     *
861:     *
862:
863:
864:     PROCEDURE newpathpop
865:     DO CASE
866:     CASE  DOS
867:        **********
868:        * Create a popup with one bar for each subdirectory
869:        * in the current path
870:        PRIVATE dirstring, dircount
```

```
871:
872:        * Base the popup on an array with one element for
873:        * each subdirectory in the current path
874:        *
875:        * Start the array with the current drive
876:        DECLARE patharray[ 1 ]
877:        patharray[ 1 ] = SET( "DEFAULT" )
878:
879:        * Get the current path string
880:        m.dirstring = CURDIR()
881:
882:        * If we are not in the root directory
883:        IF LEN( m.dirstring ) > 1
884:
885:           * Start parsing the path string for each directory name
886:           m.dircount = 1
887:
888:           * Continue as long as there is a pair of slashes
889:           * Surrounding the next part of the string
890:           DO WHILE AT( "\", m.dirstring, m.dircount ) <> 0 AND ;
891:                 AT( "\", m.dirstring, m.dircount + 1 ) <> 0
892:
893:              * Add another element at the end of the array
894:              DECLARE patharray[ m.DirCount + 1 ]
895:
896:              * Cut out the next set of characters between the slashes
897:              m.firstchar = AT( "\", m.dirstring, m.dircount ) + 1
898:
899:              m.pathlength = AT( "\", m.dirstring, m.dircount + 1 )  ;
900:                 - m.firstchar
901:
902:              * And add them to the next array element
903:              patharray[ m.DirCount + 1 ] = SUBSTR( m.dirstring, ;
904:                 m.firstchar, m.pathlength )
905:
906:              * Look for the next directory name in the path string
907:              m.dircount = m.dircount + 1
908:
909:           ENDDO
910:        ENDIF
911:
912:        * Point the popup variable at the last element in the array
913:        mpath = ALEN( patharray )
914: <------RETURN
915:
916:        ********************
917:     CASE  WINDOWS
918:        ********************
919:        * Create a popup with one bar for each subdirectory
920:        * in the current path
921:        PRIVATE dirstring, dircount
922:
923:        * Base the popup on an array with one element for
924:        * each subdirectory in the current path
925:        *
926:        * Start the array with the current drive
927:        DECLARE patharray[ 1 ]
928:        patharray[ 1 ] = SET( "DEFAULT" )
929:
930:        * Get the current path string
931:        m.dirstring = CURDIR()
932:
933:        * If we are not in the root directory
934:        IF LEN( m.dirstring ) > 1
935:
936:           * Start parsing the path string for each directory name
937:           m.dircount = 1
938:
939:           * Continue as long as there is a pair of slashes
940:           * Surrounding the next part of the string
941:           DO WHILE AT( "\", m.dirstring, m.dircount ) <> 0 AND ;
942:                 AT( "\", m.dirstring, m.dircount + 1 ) <> 0
943:
944:              * Add another element at the end of the array
945:              DECLARE patharray[ m.DirCount + 1 ]
946:
947:              * Cut out the next set of characters between the slashes
948:              m.firstchar = AT( "\", m.dirstring, m.dircount ) + 1
949:
950:              m.pathlength = AT( "\", m.dirstring, m.dircount + 1 )  ;
951:                 - m.firstchar
952:
953:              * And add them to the next array element
954:              patharray[ m.DirCount + 1 ] = SUBSTR( m.dirstring, ;
955:                 m.firstchar, m.pathlength )
956:
957:              * Look for the next directory name in the path string
958:              m.dircount = m.dircount + 1
959:
960:           ENDDO
961:        ENDIF
962:
963:        * Point the popup variable at the last element in the array
964:        mpath = ALEN( patharray )
965: <------RETURN
966:
967:        ********************
968:     ENDCASE
969:
970:     *
971:     *
972:     *              RESTORE Procedure NEWFILEPOP
973:     *
974:     *
975:     *
976:
977:
978:     PROCEDURE newfilepop
979:     DO CASE
980:     CASE  DOS
981:        ********************
982:
983:        * Create a popup with all of the file names and its subdirection i
=> n the current directory
984:        * This will be based on an array as well
985:
986:        * Create an array with all the files matching the current wild car
=> d
987:        * ADIR() creates an array with 5 columns.
988:
989:        PRIVATE startsort
990:        * Fill an array with directory names only
991:        =ADIR( dirarray, "", "D")
992:        SIZE = ALEN(dirarray,1)
993:        IF dirarray[ 1, 1 ] = "."
994:           =ADEL(dirarray,1)
995:           SIZE = SIZE - 1
996:           DECLARE dirarray[size,5]
997:        ELSE
998:           * We must be in the root directory "\"
999:           * Add one more row to the directory array
1000:          SIZE = SIZE + 1
```

```
1001:         DECLARE dirarray[ SIZE, 5 ]
1002:
1003:         * Push all the rows down by one
1004:         =AINS( dirarray, 1 )
1005:
1006:         * Fill in the first directory name in the array
1007:         * a bug in the popups makes it refuse to display a
1008:         * prompt of "\", use "\\" to make one \ appear
1009:         * even then the bar will automatically be non-selectable
1010:         * which in this case is fine
1011:         dirarray[ 1,1 ] = "\\"
1012:
1013:         * Start the sort after the root name
1014:      ENDIF
1015:
1016:      IF ALEN( dirarray, 1 ) > 2
1017:
1018:         * Sort the array starting at the starting row
1019:         =ASORT( dirarray, AELEMENT( dirarray, 2, 1 ) )
1020:      ENDIF
1021:      FOR i = 1 TO ALEN(dirarray,1)
1022:         dirarray[i,1] = "[" + dirarray[i,1] + "]"
1023:      ENDFOR
1024:
1025:      IF ADIR(filearray, m.wildcard) = 0
1026:         SIZE = ALEN(dirarray,1)
1027:         DECLARE filearray[size,5]
1028:         =ACOPY(dirarray, filearray)
1029:      ELSE
1030:         stop = ALEN(dirarray,1)
1031:         FOR i = 1 TO stop
1032:            SIZE = ALEN(filearray,1)
1033:            DECLARE filearray[size + 1,5]
1034:            =AINS(filearray,1)
1035:            filearray[1,1] = dirarray(alen[dirarray,1],1)
1036:            IF ALEN(dirarray,1) > 1
1037:               DECLARE dirarray[alen(dirarray,1)-1,5]
1038:            ENDIF
1039:         ENDFOR
1040:      ENDIF
1041:      mfile = 1
1042: <=== RETURN
1043:
1044:      ********************
1045:   CASE  WINDOWS
1046:      ********************
1047:
1048:      * Create a popup with all of the file names and its subdirection i
 => n the current directory
1049:      * This will be based on an array as well
1050:
1051:      * Create an array with all the files matching the current wild car
 => d
1052:      * ADIR() creates an array with 5 columns.
1053:
1054:      PRIVATE startsort
1055:      * Fill an array with directory names only
1056:      =ADIR( dirarray, "", "D")
1057:      SIZE = ALEN(dirarray,1)
1058:      IF dirarray[ 1, 1 ] = "."
1059:         =ADEL(dirarray,1)
1060:         SIZE = SIZE - 1
1061:         DECLARE dirarray[size,5]
1062:      ELSE
1063:         * We must be in the root directory "\"
1064:         * Add one more row to the directory array
```

```
1065:         SIZE = SIZE + 1
1066:         DECLARE dirarray[ SIZE, 5 ]
1067:
1068:         * Push all the rows down by one
1069:         =AINS( dirarray, 1 )
1070:
1071:         * Fill in the first directory name in the array
1072:         * a bug in the popups makes it refuse to display a
1073:         * prompt of "\", use "\\" to make one \ appear
1074:         * even then the bar will automatically be non-selectable
1075:         * which in this case is fine
1076:         dirarray[ 1,1 ] = "\\"
1077:
1078:         * Start the sort after the root name
1079:      ENDIF
1080:
1081:      IF ALEN( dirarray, 1 ) > 2
1082:
1083:         * Sort the array starting at the starting row
1084:         =ASORT( dirarray, AELEMENT( dirarray, 2, 1 ) )
1085:      ENDIF
1086:      FOR i = 1 TO ALEN(dirarray,1)
1087:         dirarray[i,1] = "[" + dirarray[i,1] + "]"
1088:      ENDFOR
1089:
1090:      IF ADIR(filearray, m.wildcard) = 0
1091:         SIZE = ALEN(dirarray,1)
1092:         DECLARE filearray[size,5]
1093:         =ACOPY(dirarray, filearray)
1094:      ELSE
1095:         stop = ALEN(dirarray,1)
1096:         FOR i = 1 TO stop
1097:            SIZE = ALEN(filearray,1)
1098:            DECLARE filearray[size + 1,5]
1099:            =AINS(filearray,1)
1100:            filearray[1,1] = dirarray(alen[dirarray,1],1)
1101:            IF ALEN(dirarray,1) > 1
1102:               DECLARE dirarray[alen(dirarray,1)-1,5]
1103:            ENDIF
1104:         ENDFOR
1105:      ENDIF
1106:      mfile = 1
1107: <=== RETURN
1108:
1109:      ********************
1110:   ENDCASE
1111:
1112:      *
1113:      *
1114:      *          RESTORE Function PATHSTRING
1115:      *
1116:      *
1117:      *
1118:
1119:
1120:   PROCEDURE pathstring
1121:   DO CASE
1122:   CASE  DOS
1123:      ********************
1124:      * Convert an array of subdirectories, such as PathArray,
1125:      * into a legal path string for use with SET DEFAULT
1126:      * Use the current value of the path popup as the
1127:      * ending point on the path
1128:      PRIVATE ppath
1129:
1130:      * Start with the drive name
```

```
1131:       m.ppath = patharray[ 1 ]
1132:
1133:       * If the path popup is pointing to a subdirectory
1134:      IF mpath > 1
1135:
1136:         * Add all the subdirectories to the path string
1137:        FOR i = 2 TO mpath
1138:           m.ppath = m.ppath + "\" + patharray[ i ]
1139:
1140:        NEXT
1141:      ENDIF
1142:
1143:       * End the path with one last slash for good luck
1144:       m.ppath = m.ppath + "\"
1145:
1146: <-----RETURN m.ppath
1147:
1148:       ********************
1149:    CASE  WINDOWS
1150:       ********************
1151:       * Convert an array of subdirectories, such as PathArray,
1152:       * into a legal path string for use with SET DEFAULT
1153:       * Use the current value of the path popup as the
1154:       * ending point on the path
1155:       PRIVATE ppath
1156:
1157:       * Start with the drive name
1158:       m.ppath = patharray[ 1 ]
1159:
1160:       * If the path popup is pointing to a subdirectory
1161:      IF mpath > 1
1162:
1163:         * Add all the subdirectories to the path string
1164:        FOR i = 2 TO mpath
1165:           m.ppath = m.ppath + "\" + patharray[ i ]
1166:
1167:        NEXT
1168:      ENDIF
1169:
1170:       * End the path with one last slash for good luck
1171:       m.ppath = m.ppath + "\"
1172:
1173: <-----RETURN m.ppath
1174:
1175:       ********************
1176:    ENDCASE
1177:
1178:     *
1179:     *
1180:     *                RESTORE Procedure PKUNZIP
1181:     *
1182:     *
1183:     *
1184:
1185:
1186:    PROCEDURE pkunzip
1187:    DO CASE
1188:    CASE  DOS
1189:       ********************
1190:
1191:       PRIVATE CURDIR, datadir, msel
1192:       CURDIR = FULLPATH(CURDIR())
1193:      IF !EMPTY(GETENV("CAMD"))
1194:         datadir = FULLPATH(GETENV("CAMD"))
1195:      ELSE
1196:         datadir = CURDIR()
1197:      ENDIF
1198:       desdir = pathstring() + mfname
1199:       SET DEFA TO (datadir)
1200:      IF !EMPTY(GETENV("CAMDUTIL"))
1201:         pkzipcom = "!" + GETENV("CAMDUTIL") + "\PKUNZIP -O &desdir *.db
 => f *.cdx *.idx *.fpt"
1202:      ELSE
1203:         pkzipcom = "!PKUNZIP -O &desdir *.dbf *.cdx *.idx *.fpt"
1204:      ENDIF
1205:       &pkzipcom
1206:       USE
1207:       DELETE FILE t0000000.txt
1208:       SET DEFA TO (CURDIR)
1209:
1210: <-----RETURN
1211:    CASE  WINDOWS
1212:       ********************
1213:
1214:       PRIVATE CURDIR, datadir, msel
1215:       CURDIR = FULLPATH(CURDIR())
1216:      IF !EMPTY(GETENV("KBDATA"))
1217:         datadir = FULLPATH(GETENV("KBDATA"))
1218:      ELSE
1219:         datadir = CURDIR()
1220:      ENDIF
1221:       desdir = pathstring() + mfname
1222:       SET DEFA TO (datadir)
1223:      IF !EMPTY(GETENV("CAMDUTIL"))
1224:         pkzipcom = "!" + GETENV("CAMDUTIL") + "\PKUNZIP -O &desdir *.db
 => f *.cdx *.idx *.fpt"
1225:      ELSE
1226:         pkzipcom = "!PKUNZIP -O &desdir *.dbf *.cdx *.idx *.fpt"
1227:      ENDIF
1228:       &pkzipcom
1229:       USE
1230:       DELETE FILE t0000000.txt
1231:       SET DEFA TO (CURDIR)
1232:
1233: <-----RETURN
1234:    ENDCASE
1235:  *:  EOF: RESTORE.ac1
```

```
*
*       ┌──────────────────────────────────────────────────────┐
*       │ 08/09/94              DD.SPR               09:39:55  │
*       │                                                      │
*       ├──────────────────────────────────────────────────────┤
*       │                                                      │
*       │ Author's Name                                        │
*       │                                                      │
*       │ Copyright (c) 1994 Company Name                      │
*       │ Address                                              │
*       │ City,     Zip                                        │
*       │                                                      │
*       │ Description:                                         │
*       │ This program was automatically generated by GENSCRN. │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘


#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET readborder ON

*
*       ┌──────────────────────────────────────────────────────┐
*       │             Windows Window definitions               │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

IF NOT WEXIST("w_dd") ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.PJX" ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.SCX" ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.MNX" ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.PRG" ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.FRX" ;
      OR UPPER(WTITLE("W_DD")) == "W_DD.QPR"
   DEFINE WINDOW w_dd ;
      AT  0.000, 0.000 ;
      SIZE 30.000,99.200 ;
      TITLE "Question - Data Dictionary" ;
      FONT "MS Sans Serif", 8 ;
      FLOAT ;
      NOCLOSE ;
      MINIMIZE ;
      SYSTEM
   MOVE WINDOW w_dd CENTER
ENDIF


*
*       ┌──────────────────────────────────────────────────────┐
*       │         DD/Windows Setup Code - SECTION 2            │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

#REGION 1
SELECT 0
USE area INDEX area
SELECT 0
USE enum INDEX enum
SELECT 0
USE VAL INDEX VAL
SELECT 0
USE dict INDEX dictid, dictname
SET RELATION TO id INTO enum, id INTO VAL
SELECT 0
USE goals INDEX goals
SELECT 0
USE quals INDEX quals
SELECT dict
SET ORDER TO 2
COPY TO ARRAY mdict FIELDS name, id
mq = 1


*
*       ┌──────────────────────────────────────────────────────┐
*       │              DD/Windows Screen Layout                │
*       │                                                      │
*       └──────────────────────────────────────────────────────┘
*

#REGION 1
IF WVISIBLE("w_dd")
   ACTIVATE WINDOW w_dd SAME
ELSE
   ACTIVATE WINDOW w_dd NOSHOW
ENDIF
@ 28.077,25.600 GET mbuttons ;
   PICTURE "@*HN \<Edit;\<Add;\<Delete;\<Quit" ;
   SIZE 1.917,7.500,0.750 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpt2i()
@ 1.077,7.400 GET mq ;
   PICTURE "@&N" ;
   FROM mdict ;
   SIZE 26.833,50.625 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kpt7d()

IF NOT WVISIBLE("w_dd")
   ACTIVATE WINDOW w_dd
ENDIF

READ CYCLE MODAL

RELEASE WINDOW w_dd

#REGION 0

SET readborder &rborder

IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF
```

```
133:
134:
135:     *
136:     *
137:     *                    DD/Windows Cleanup Code
138:     *
139:     *
140:     *
141:
142:     #REGION 1
143:     SELECT area
144:     USE
145:     SELECT enum
146:     USE
147:     SELECT VAL
148:     USE
149:     SELECT dict
150:     USE
151:     SELECT goals
152:     USE
153:     SELECT quals
154:     USE
155: <---RETURN
156:
157:     *-----------------------------------
158:     *
159:     * This procedure validate the object to have access to delete or not.
160:     *
161:     *-----------------------------------
162:
163:
164:     *
165:     *
166:     *                DD/Windows Supporting Procedures and Functions
167:     *
168:     *
169:     *
170:
171:     #REGION 1
172:     FUNCTION validobj
173:     PRIVATE msel,mvalid
174:     mvalid = .F.
175:     msel = SELECT()
176:     SELECT quals
177:     SEEK mdict[mq,2]
178:    ┌IF FOUND()
179:    │  ┌IF EMPTY(rules) AND EMPTY(ruleso)
180:    │  │    mvalid = .T.
181:    │  ├ELSE
182:    │  │    =errmsg("Delete was not allow, the object was occupied",2)
183:    │  └ENDIF
184:    ├ELSE
185:    │    mvalid = .T.
186:    └ENDIF
187:     SELECT (msel)
188: <---RETURN mvalid
189:
190:     FUNCTION qualdel
191:     PRIVATE msel
192:     msel = SELECT()
193:     SELECT dict
194:     SET ORDER TO 1
195:     SEEK mdict[mq,2]
196:    ┌IF FOUND()
197:    │    = popupshow("deleting...")
198:    │  ┌DO CASE
199:    │  ├CASE dict.datatype $ "ELM"
200:    │  │    SELECT enum
201:    │  │    DELETE FOR id = dict.id
202:    │  │    PACK
203:    │  ├CASE dict.datatype $ "N"
204:    │  │    SELECT VAL
205:    │  │    DELETE FOR id = dict.id
206:    │  │    PACK
207:    │  └ENDCASE
208:    │    SELECT dict
209:    │    DELETE
210:    │    PACK
211:    │    = popuphide()
212:    └ENDIF
213:     SELECT (msel)
214: <---RETURN
215:
216:     FUNCTION resetdata
217:     PRIVATE msel
218:     msel = SELECT()
219:     SELECT dict
220:     SET ORDER TO 2
221:     COPY TO ARRAY mdict FIELDS name, id
222:     SELECT (msel)
223: <---RETURN
224:
225:     *
226:     *
227:     *    _QSF0KPT2I            mbuttons VALID
228:     *
229:     *    Function Origin:
230:     *
231:     *    From Platform:        Windows
232:     *    From Screen:          DD,     Record Number:    2
233:     *    Variable:             mbuttons
234:     *    Called By:            VALID Clause
235:     *    Object Type:          Push Button
236:     *    Snippet Number:       1
237:     *
238:     *
239:     *
240:     FUNCTION _qsf0kpt2i     &&  mbuttons VALID
241:     #REGION 1
242:    ┌DO CASE
243:    ├CASE mbuttons = 1
244:    │  ┌IF mq > 0
245:    │  │    DO ddedit.spr WITH mdict[mq,2], mdict[mq,1]
246:    │  └ENDIF
247:    ├CASE mbuttons = 2   && add
248:    │    m.adding = .T.
249:    │    SELECT dict
250:    │    SET ORDER TO 1
251:    │    GOTO BOTTOM
252:    │    m.id = id + 1
253:    │    m.name = ""
254:    │    DO ddedit.spr WITH m.id, m.name
255:    ├CASE mbuttons = 3   && delete
256:    │    mdel = validobj()
257:    │  ┌IF mdel
258:    │  │  . = qualdel()
259:    │  └ENDIF
260:    ├CASE mbuttons = 4   && Quit
261:    │    mok = .T.
262:    │    CLEAR READ
263:    └ENDCASE
264:     DO resetdata
```

```
265:     SHOW GETS
266:
267:     *
268:     *
269:     *          _QSF0KPT7D            mq VALID
270:     *
271:     *          Function Origin:
272:     *
273:     *          From Platform:        Windows
274:     *          From Screen:          DD,     Record Number:    3
275:     *          Variable:             mq
276:     *          Called By:            VALID Clause
277:     *          Object Type:          List
278:     *          Snippet Number:       2
279:     *
280:     *
281:     *
282:     FUNCTION _qsf0kpt7d     &&  mq VALID
283:     #REGION 1
284:    ┌IF mq > 0
285:    │   DO ddedit.spr WITH mdict[mq,2], mdict[mq,1]
286:    └ENDIF
287:  *:  EOF: DD.ac1
```

```
*
*       08/09/94              PLAN.SPR                  09:39:58
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

DO CASE
CASE _WINDOWS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              Windows Window definitions          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("w_goal") ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PJX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.SCX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.MNX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PRG" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.FRX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.QPR"
      DEFINE WINDOW w_goal ;
         AT  0.000, 0.000  ;
         SIZE 19.000,64.000 ;
         TITLE "Goal Object Editor" ;
         FONT "MS Sans Serif", 8 ;
         STYLE "B" ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE
      MOVE WINDOW w_goal CENTER
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              PLAN/Windows Screen Layout          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal SAME
   ELSE
      ACTIVATE WINDOW w_goal NOSHOW
   ENDIF
   @ 16.769,17.000 GET mbuttons ;
      PICTURE "@*HN \<Add;\<OK;\<Cancel" ;
      SIZE 1.769,8.000,1.000 ;
      DEFAULT 1 ;
      FONT "MS Sans Serif", 8 ;
      STYLE "B" ;
      VALID _qsf0kpv8r()
   @ 0.462,2.500 GET mg ;
      PICTURE "@&N" ;
      FROM mgoals ;
      SIZE 15.000,71.000 ;
      DEFAULT 1 ;
      FONT "MS Sans Serif", 8 ;
      VALID _qsf0kpvcr()

   IF NOT WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_goal

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


CASE _DOS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
```

```
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   *
   *
   *              MS-DOS Window definitions
   *
   *
   *

   IF NOT WEXIST("w_goal") ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PJX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.SCX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.MNX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.PRG" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.FRX" ;
         OR UPPER(WTITLE("W_GOAL")) == "W_GOAL.QPR"
      DEFINE WINDOW w_goal ;
         FROM INT((SROW()-19)/2),INT((SCOL()-64)/2) ;
         TO INT((SROW()-19)/2)+18,INT((SCOL()-64)/2)+63 ;
         TITLE "Goal Object Editor" ;
         FLOAT ;
         CLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         COLOR SCHEME 1
   ENDIF


   *
   *
   *              PLAN/MS-DOS Screen Layout
   *
   *
   *

   #REGION 1
   IF WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal SAME
   ELSE
      ACTIVATE WINDOW w_goal NOSHOW
   ENDIF
   @ 16,17 GET mbuttons ;
      PICTURE "@*HN \<Add;\<OK;\<Cancel" ;
      SIZE 1,8,1 ;
      DEFAULT 1 ;
      VALID _qsf0kpvqb()
   @ 1,1 GET mg ;
      PICTURE "@&N" ;
      FROM mgoals ;
      SIZE 14,60 ;
      DEFAULT 1 ;
      VALID _qsf0kpvw8() ;
      COLOR SCHEME 2

   IF NOT WVISIBLE("w_goal")
      ACTIVATE WINDOW w_goal
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW w_goal

   #REGION 0
   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


ENDCASE


    *
    *
    *       _QSF0KPV8R          mbuttons VALID
    *
    *       Function Origin:
    *
    *       From Platform:      Windows
    *       From Screen:        PLAN,     Record Number:   2
    *       Variable:           mbuttons
    *       Called By:          VALID Clause
    *       Object Type:        Push Button
    *       Snippet Number:     1
    *
    *
    *
    FUNCTION _qsf0kpv8r     &&  mbuttons VALID
    #REGION 1
    DO CASE
    CASE mbuttons = 1
       DEACTIVATE WINDOW w_goal
       DO object.spr WITH "G"
       ACTIVATE WINDOW w_goal
       DO setgoals
    CASE mbuttons = 2   && ok
       mok = .T.
       CLEAR READ
    CASE mbuttons = 3   && cancel
       mok = .F.
       CLEAR READ
    ENDCASE
    SHOW GETS


    *
    *
    *       _QSF0KPVCR          mg VALID
    *
    *       Function Origin:
    *
    *       From Platform:      Windows
    *       From Screen:        PLAN,     Record Number:   3
    *       Variable:           mg
    *       Called By:          VALID Clause
    *       Object Type:        List
    *       Snippet Number:     2
```

```
*
*
*
FUNCTION _qsf0kpvcr     &&  mg VALID
#REGION 1
DO edgoal

*
*
*       _QSFOKPVQB           mbuttons VALID
*
*       Function Origin:
*
*       From Platform:       MS-DOS
*       From Screen:         PLAN,     Record Number:    7
*       Variable:            mbuttons
*       Called By:           VALID Clause
*       Object Type:         Push Button
*       Snippet Number:      3
*
*
*
FUNCTION _qsf0kpvqb     &&  mbuttons VALID
#REGION 1
DO CASE
CASE mbuttons = 1
   DEACTIVATE WINDOW w_goal
   DO object.spr WITH "G"
   ACTIVATE WINDOW w_goal
   DO setgoals
CASE mbuttons = 2   && ok
   mok = .T.
   CLEAR READ
CASE mbuttons = 3   && cancel
   mok = .F.
   CLEAR READ
ENDCASE
SHOW GETS


*
*
*       _QSFOKPVW8           mg VALID
*
*       Function Origin:
*
*       From Platform:       MS-DOS
*       From Screen:         PLAN,     Record Number:    8
*       Variable:            mg
*       Called By:           VALID Clause
*       Object Type:         List
*       Snippet Number:      4
*
*
*
FUNCTION _qsf0kpvw8     &&  mg VALID
#REGION 1
DO edgoal
*:  EOF: PLAN.ac1
```

```
  1:  *:*************************************************************************
=> *******
  2:  *:
  3:  *: Procedure file: C:\CAMD2\KBEDIT\WORKW\ERRMSG.PRG
  4:  *:         System: Knowledge Base Editor
  5:  *:         Author: Hoa L. Ly
  6:  *:      Copyright (c) June 1, 1994, Naval Health Research Center, Code 2
=> 2
  7:  *:  Last modified: 07/29/94 at  9:22:58
  8:  *:
  9:  *:         Set by: ACTION.SPR
 10:  *:               : ACTELSE.SPR
 11:  *:               :  QUIT()           (function in KBINIT.PRG)
 12:  *:               : VALIDOBJ()        (function in QUAL.SPR)
 13:  *:
 14:  *:      Documented 15:01:10                                  FoxDoc versio
=> n 3.00a
 15:  *:*************************************************************************
=> *******
 16:
 17:     * Open a message window
 18:     *
 19:     * ERRMSG( <expC>[, <expN>] )
 20:     *
 21:     PARAMETERS errmsg, timelimit
 22:     PRIVATE ALL
 23:     * If no errmessage is sent then quit
 24:   ┌─IF PARAMETERS()=0
 25: <═══RETURN
 26:   └─ENDIF
 27:
 28:     * m.TimeLimit is used in a WAIT TIMEOUT command
 29:     * to control the amount of time ERRMSG display's its message
 30:     * If no time limit is received set time to 0 wait forever
 31:
 32:   ┌─IF PARAMETERS()=1 OR EMPTY(m.timelimit)
 33:   │   m.timelimit=0
 34:   └─ENDIF
 35:
 36:   ┌─IF TYPE("m.TimeLimit")!="N"
 37:   │   m.timelimit=IIF(TYPE("m.TimeLImit")="C",INT(VAL(m.timelimit)),0)
 38:   └─ENDIF
 39:
 40:     m.errmsg=IIF(TYPE("m.errmsg")!="C","",m.errmsg)
 41:
 42:   ┌─IF PARAMETERS()=2 AND EMPTY(m.errmsg) AND m.timelimit>0
 43: <═══RETURN
 44:   └─ENDIF
 45:
 46:     * set talk off
 47:   ┌─IF SET('TALK') = 'ON'  && TALK handled as a special case.
 48:   │   SET TALK OFF          && Turn TALK OFF
 49:   │   savetalk = 'ON'       && TALK was ON, save the setting
 50:   ├─ELSE   && TALK is OFF
 51:   │   savetalk = 'OFF'      && TALK was OFF, save the setting
 52:   └─ENDIF
 53:
 54:     * Get the length of the message to size window
 55:     m.len=LEN(m.errmsg)
 56:     m.high = 1
 57:     m.old = SET("memowidth")
 58:     SET MEMOWIDTH TO 75
 59:     * Set miminum length for the Press any key
 60:   ┌─IF m.timelimit = 0
 61:   │   m.len=IIF(m.len<30,30,m.len)
 62:   └─ENDIF
 63:
 64:   ┌─IF m.len >75
 65:   │   m.high=MEMLINES(m.errmsg)
 66:   │   m.len=75
 67:   └─ENDIF
 68:
 69:     *Find begining/ending of the window
 70:     m.begin=40-(INT(m.len/2)+1)
 71:     m.end=40+(INT(m.len/2)+1)
 72:
 73:     * Remember the current window status
 74:     m.oldwindow =IIF( WOUTPUT() = errmsg, "" , WOUTPUT() )
 75:
 76:     * Create the window
 77:     DEFINE WINDOW errmsg ;
 78:        AT 22, BEGIN;
 79:        SIZE m.high + 3, m.len + 5 ;
 80:        FONT "Terminal", 8 ;
 81:        FLOAT ;
 82:        NOCLOSE ;
 83:        MINIMIZE ;
 84:        DOUBLE ;
 85:        COLOR RGB(,,,0,0,255)
 86:
 87:     *define window errmsg from 18,m.begin to 23,m.end color scheme 1
 88:     ACTIVATE WINDOW errmsg
 89:
 90:     * Print the message centered in the window
 91:
 92:     CLEAR
 93:     @ 1, ( WCOL() - m.len )/2 SAY m.errmsg ;
 94:        SIZE (m.high), (m.len);
 95:        FONT "Terminal",8
 96:
 97:   ┌─IF m.timelimit = 0
 98:   │   *   m.presskey = "Press any key to continue"
 99:   │   *   @ 2, ( wcol() - len( m.presskey ) )/2 say m.presskey
100:   │ ┌─IF m.high > 1
101:   │ │   @ ROW(), 1 SAY ""
102:   │ ├─ELSE
103:   │ │   @ ROW() + 1, 1 SAY ""
104:   │ └─ENDIF
105:   │   WAIT
106:   ├─ELSE
107:   │   *   Wait for the number of seconds in m.Timeout
108:   │   *   A value of 0 will wait forever
109:   │   WAIT "" TIMEOUT m.timelimit
110:   └─ENDIF
111:
112:
113:     * Close Window
114:     RELEASE WINDOW errmsg
115:
116:     * If there was no output window originally
117:   ┌─IF EMPTY( m.oldwindow )
118:   │
119:   │   * Send future output back to the screen
120:   │   ACTIVATE SCREEN
121:   │
122:   ├─ELSE
123:   │
124:   │   * Return output to the original window
125:   │   ACTIVATE WINDOW ( m.oldwindow )
126:   └─ENDIF
127:
128:    SET TALK &savetalk   && Restore original TALK setting
```

X-1

ICA/4PRINT

```
129: <——RETURN
130:  *:  EOF: ERRMSG.act
```

```
*
*      08/09/94              DICT.SPR                  09:40:02
*
*      Author's Name
*
*      Copyright (c) 1994 Company Name
*      Address
*      City,     Zip
*
*      Description:
*      This program was automatically generated by GENSCRN.
*
*

PARAMETERS m.newid

*
*            DICT/Windows Setup Code - SECTION 1
*
*

#REGION 1
PRIVATE mp, mpn, msel, m.adding
DIMENSION enum[20], enumr[20]
msel = SELECT()
SELECT dict
enum = " "
enumr = 0
mp = 1
m.adding = .F.
m.qrecno = RECNO()
SEEK m.newid
IF !FOUND()
   m.adding = .T.
   SCATTER MEMVAR BLANK
   m.id = m.newid
   SHOW GET m.datatype ENABLE
ELSE
   SCATTER MEMVAR
ENDIF
DO setenum
*do setval

#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET readborder ON

*
*            Windows Window definitions
*
```

```
*
*
*

IF NOT WEXIST("dict") ;
      OR UPPER(WTITLE("DICT")) == "DICT.PJX" ;
      OR UPPER(WTITLE("DICT")) == "DICT.SCX" ;
      OR UPPER(WTITLE("DICT")) == "DICT.MNX" ;
      OR UPPER(WTITLE("DICT")) == "DICT.PRG" ;
      OR UPPER(WTITLE("DICT")) == "DICT.FRX" ;
      OR UPPER(WTITLE("DICT")) == "DICT.QPR"
   DEFINE WINDOW dict ;
      AT  0.000, 0.000  ;
      SIZE 26.250,68.875 ;
      TITLE "Dictionary Editor" ;
      FONT "Terminal", 8 ;
      FLOAT ;
      NOCLOSE ;
      MINIMIZE ;
      SYSTEM ;
      COLOR RGB(,,,0,255,255)
   MOVE WINDOW dict CENTER
ENDIF

*
*            DICT/Windows Screen Layout
*
*

#REGION 1
IF WVISIBLE("dict")
   ACTIVATE WINDOW dict SAME
ELSE
   ACTIVATE WINDOW dict NOSHOW
ENDIF
@ 1.333,55.250 SAY "Type"  ;
   FONT "Terminal", 8
@ 22.167,10.500 SAY "Hi"  ;
   FONT "Terminal", 8
@ 22.167,30.000 SAY "Lo"  ;
   FONT "Terminal", 8
@ 22.167,48.000 SAY "Units"  ;
   FONT "Terminal", 8
@ 1.333,3.000 SAY "id"  ;
   FONT "Terminal", 8
@ 1.417,12.000 SAY "Name"  ;
   FONT "Terminal", 8
@ 20.167,9.000 SAY "Width"  ;
   FONT "Terminal", 8
@ 20.167,30.000 SAY "Decimals"  ;
   FONT "Terminal", 8
@ 22.083,3.000 SAY "Range:"  ;
   FONT "Terminal", 8
@ 8.667,3.375 SAY "Enum"  ;
   FONT "Terminal", 8
@ 20.167,3.000 SAY "Val:"  ;
   FONT "Terminal", 8
@ 1.167,6.250 GET m.id ;
   SIZE 1.000,3.125 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   DISABLE
@ 1.167,18.250 GET m.name ;
```

```
133:        SIZE 1.000,33.875 ;
134:        DEFAULT " " ;
135:        FONT "Terminal", 8
136:     @ 1.083,60.500 GET m.datatype ;
137:        PICTURE "@^ N;E;M;L" ;
138:        SIZE 1.500,4.375 ;
139:        DEFAULT "N" ;
140:        FONT "Terminal", 8 ;
141:        VALID _qsf0kpy4x() ;
142:        DISABLE
143:     @ 3.167,21.250 GET dict.askable ;
144:        SIZE 1.000,4.625 ;
145:        DEFAULT .F. ;
146:        FONT "Terminal", 8
147:     @ 5.000,3.250 EDIT m.question ;
148:        SIZE 3.000,61.750,0.000 ;
149:        DEFAULT " " ;
150:        FONT "Terminal", 8 ;
151:        SCROLL ;
152:        WHEN dict.askable
153:     @ 10.167,3.125 GET mp ;
154:        PICTURE "@&N" ;
155:        FROM enum ;
156:        SIZE 9.333,62.000 ;
157:        DEFAULT 1 ;
158:        FONT "Terminal", 8 ;
159:        WHEN m.datatype $ "EML" ;
160:        VALID _qsf0kpy8w()
161:     @ 20.167,16.750 GET m.width ;
162:        SIZE 1.000,5.500 ;
163:        DEFAULT 0 ;
164:        FONT "Terminal", 8 ;
165:        WHEN m.datatype = "N" ;
166:        DISABLE
167:     @ 20.167,39.250 GET m.dec ;
168:        SIZE 1.000,11.500 ;
169:        DEFAULT 0 ;
170:        FONT "Terminal", 8 ;
171:        WHEN m.datatype = "N" ;
172:        DISABLE
173:     @ 22.167,16.750 GET m.hi ;
174:        SIZE 1.000,10.000 ;
175:        DEFAULT 0 ;
176:        FONT "Terminal", 8 ;
177:        WHEN m.datatype = "N" ;
178:        DISABLE
179:     @ 22.167,33.250 GET m.lo ;
180:        SIZE 1.000,13.000 ;
181:        DEFAULT 0 ;
182:        FONT "Terminal", 8 ;
183:        WHEN m.datatype = "N" ;
184:        DISABLE
185:     @ 22.167,54.250 GET m.units ;
186:        SIZE 1.000,10.000 ;
187:        DEFAULT " " ;
188:        FONT "Terminal", 8 ;
189:        WHEN m.datatype = "N" ;
190:        DISABLE
191:     @ 24.000,25.500 GET mbuttons ;
192:        PICTURE "@*HT OK;Cancel" ;
193:        SIZE 1.917,8.375,0.750 ;
194:        DEFAULT 1 ;
195:        FONT "Terminal", 8 ;
196:        VALID _qsf0kpyf6()
197:     @ 3.167,3.000 SAY "Question Askable:"  ;
198:        FONT "Terminal", 8 ;
199:        STYLE "T"
200:
201:  ┌─IF NOT WVISIBLE("dict")
202:  │     ACTIVATE WINDOW dict
203:  └─ENDIF
204:
205:    READ CYCLE MODAL ;
206:       WHEN _qsf0kpyj0()
207:
208:    RELEASE WINDOW dict
209:
210:    #REGION 0
211:
212:    SET readborder &rborder
213:
214:  ┌─IF m.talkstat = "ON"
215:  │     SET TALK ON
216:  └─ENDIF
217:  ┌─IF m.compstat = "ON"
218:  │    . SET COMPATIBLE ON
219:  └─ENDIF
220:
221:
222:    *
223:    *
224:    *                    DICT/Windows Cleanup Code
225:    *
226:    *
227:    *
228:
229:    #REGION 1
230:    SELECT dict
231:    GOTO m.qrecno
232: <──RETURN
233:
234:
235:
236:    *
237:    *
238:    *          DICT/Windows Supporting Procedures and Functions
239:    *
240:    *
241:    *
242:
243:    #REGION 1
244:    PROCEDURE edenum
245:    SELECT enum
246:  ┌─IF enumr[mp] > 0
247:  │     GOTO enumr[mp]
248:  └─ENDIF
249:    *m.id = dict.id
250:    DO enum.spr
251:    DO setenum
252:    SHOW GETS
253:    SELECT dict
254: <──RETURN
255:
256:
257:    PROCEDURE setenum
258:    enum = " "
259:    enumr = 0
260:  ┌─IF m.datatype $ "EML"
261:  │     PRIVATE msel, i
262:  │     msel = SELECT()
263:  │     SELECT enum
264:  │     SEEK m.id
```

```
265:        i = 0
266:        DO WHILE id = m.id
267:           i = i + 1
268:           enumr[i] = RECNO()
269:           enum[i] = mutex + " " + TRIM( enumerate )
270:           SKIP
271:        ENDDO
272:        mpn = i
273:        mp = MIN(mpn, mp)
274:        mp = MAX(1, mp)
275:        IF enumr[mp] > 0
276:           GOTO enumr[mp]
277:        ENDIF
278:        SELECT (msel)
279:     ENDIF
280: <==RETURN
281:
282:     PROCEDURE setval
283:     IF m.datatype $ "N"
284:        PRIVATE msel, i
285:        msel = SELECT()
286:        SELECT VAL
287:        SCATTER MEMVAR
288:        SHOW GETS
289:        SELECT (msel)
290:     ENDIF
291: <==RETURN
292:
293:
294:     *
295:     *
296:     *        _QSF0KPY4X            m.datatype VALID
297:     *
298:     *        Function Origin:
299:     *
300:     *        From Platform:        Windows
301:     *        From Screen:          DICT,     Record Number:   15
302:     *        Variable:             m.datatype
303:     *        Called By:            VALID Clause
304:     *        Object Type:          Popup
305:     *        Snippet Number:       1
306:     *
307:     *
308:     *
309:     FUNCTION _qsf0kpy4x     &&  m.datatype VALID
310:     #REGION 1
311:     SHOW GET m.datatype DISABLE
312:
313:     *
314:     *
315:     *        _QSF0KPY8W            mp VALID
316:     *
317:     *        Function Origin:
318:     *
319:     *        From Platform:        Windows
320:     *        From Screen:          DICT,     Record Number:   18
321:     *        Variable:             mp
322:     *        Called By:            VALID Clause
323:     *        Object Type:          List
324:     *        Snippet Number:       2
325:     *
326:     *
327:     *
328:     FUNCTION _qsf0kpy8w     &&  mp VALID
329:     #REGION 1
330:     DO edenum
```



```
331:
332:     *
333:     *
334:     *        _QSF0KPYF6            mbuttons VALID
335:     *
336:     *        Function Origin:
337:     *
338:     *        From Platform:        Windows
339:     *        From Screen:          DICT,     Record Number:   24
340:     *        Variable:             mbuttons
341:     *        Called By:            VALID Clause
342:     *        Object Type:          Push Button
343:     *        Snippet Number:       3
344:     *
345:     *
346:     *
347:     FUNCTION _qsf0kpyf6     &&  mbuttons VALID
348:     #REGION 1
349:     DO CASE
350:     CASE mbuttons = 1   && ok
351:        SELECT dict
352:        GATHER MEMVAR
353:        IF m.datatype $ "N"
354:           SELECT VAL
355:           GATHER MEMVAR
356:        ENDIF
357:     CASE mbuttons = 2   && cancel
358:        mok = .F.
359:     ENDCASE
360:
361:     *
362:     *
363:     *        _QSF0KPYJ0            Read Level When
364:     *
365:     *        Function Origin:
366:     *
367:     *
368:     *        From Platform:        Windows
369:     *        From Screen:          DICT
370:     *        Called By:            READ Statement
371:     *        Snippet Number:       4
372:     *
373:     *
374:     *
375:     FUNCTION _qsf0kpyj0     && Read Level When
376:     *
377:     * When Code from screen: DICT
378:     *
379:     #REGION 1
380:     IF m.datatype $ "N"
381:        PRIVATE msel
382:        msel = SELECT()
383:        SELECT VAL
384:        SCATTER MEMVAR
385:        SHOW GET mp DISABLE
386:        SHOW GET m.width ENABLE
387:        SHOW GET m.dec ENABLE
388:        SHOW GET m.hi ENABLE
389:        SHOW GET m.lo ENABLE
390:        SHOW GET m.units ENABLE
391:        SHOW GETS
392:     ENDIF
393: *:  EOF: DICT.ac1
```

```
*
*       08/09/94            KBEDIT.SPR            09:40:05
*
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*


#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET readborder ON

*
*
*                 Windows Window definitions
*
*
*

IF NOT WEXIST("kbedit") ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.PJX" ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.SCX" ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.MNX" ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.PRG" ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.FRX" ;
      OR UPPER(WTITLE("KBEDIT")) == "KBEDIT.QPR"
   DEFINE WINDOW kbedit ;
      AT  0.000, 0.000 ;
      SIZE 29.667,64.250 ;
      TITLE "Knowledge Base Rule Editor" ;
      FONT "Terminal", 8 ;
      FLOAT ;
      NOCLOSE ;
      MINIMIZE ;
      SYSTEM ;
      COLOR RGB(,,,128,128,128)
   MOVE WINDOW kbedit CENTER
ENDIF


*
*
*          KBEDIT/Windows Setup Code - SECTION 2
*
*
*

#REGION 1
PRIVATE mp, me, ma, mpn, man, ntriples,ptop
DIMENSION prem[50], premr[50], triple[50,5]
DIMENSION premo[50], prems[50], premm[50]
DIMENSION act[50], actr[50], actelse[50], actelser[50]

PRIVATE msel
msel = SELECT()
SELECT 0
USE fact INDEX fact
SELECT 0
USE premise INDEX premise
SET RELATION TO fact INTO fact
SELECT 0
USE action INDEX action
SELECT 0
USE rule INDEX rulearea,salience,rule
SET RELATION TO premise INTO premise, action INTO action
SELECT area
SET RELATION TO area INTO rule
SELECT rule
SET TOPIC TO "RULE"
mp   = 1
act  = " "
actr = 0
ma   = 1
me = 1
DO setprem
DO setact
DO setelse
DO setaction



*
*
*              KBEDIT/Windows Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("kbedit")
   ACTIVATE WINDOW kbedit SAME
ELSE
   ACTIVATE WINDOW kbedit NOSHOW
ENDIF
@ 3.167,1.625 SAY "Rule:"  ;
   FONT "Terminal", 8 ;
   STYLE "T"
@ 1.167,18.250 GET m.areaname ;
   SIZE 1.000,34.000 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   DISABLE
@ 3.167,7.750 GET rule.rule ;
   SIZE 1.000,25.000 ;
   DEFAULT 0 ;
   FONT "Terminal", 8 ;
   DISABLE
@ 5.000,1.500 GET minvprem ;
   PICTURE "@*HN \<IF:" ;
   SIZE 1.500,7.250,0.750 ;
   DEFAULT 1 ;
```

```
133:        FONT "Terminal", 8 ;
134:        VALID _qsf0kq0sa() ;
135:        MESSAGE "Premises"
136:     @ 7.167,1.625 GET mp ;
137:        PICTURE "@&N" ;
138:        FROM prem ;
139:        SIZE 4.667,50.875 ;
140:        DEFAULT 1 ;
141:        FONT "Terminal", 8
142:     @ 13.000,1.500 GET minvact ;
143:        PICTURE "@*HN \<THEN:" ;
144:        SIZE 1.583,7.125,0.750 ;
145:        DEFAULT 1 ;
146:        FONT "Terminal", 8 ;
147:        VALID _qsf0kq0wk() ;
148:        MESSAGE "Actions"
149:     @ 15.167,1.625 GET ma ;
150:        PICTURE "@&N" ;
151:        FROM act ;
152:        SIZE 4.667,50.875 ;
153:        DEFAULT 1 ;
154:        FONT "Terminal", 8
155:     @ 21.000,1.500 GET minvelse ;
156:        PICTURE "@*HN \<ELSE:" ;
157:        SIZE 1.583,7.250,0.750 ;
158:        DEFAULT 1 ;
159:        FONT "Terminal", 8 ;
160:        VALID _qsf0kq10r() ;
161:        MESSAGE "Actions (Else)"
162:     @ 23.167,1.625 GET me ;
163:        PICTURE "@&N" ;
164:        FROM actelse ;
165:        SIZE 4.667,50.875 ;
166:        DEFAULT 1 ;
167:        FONT "Terminal", 8
168:     @ 1.000,54.000 GET mbbutton ;
169:        PICTURE "@*VN \<Next;\<Prev;\<Browse;\<Edit;e\<Xit" ;
170:        SIZE 1.750,8.875,1.083 ;
171:        DEFAULT 1 ;
172:        FONT "Terminal", 8 ;
173:        VALID _qsf0kq14z()
174:     @ 1.167,1.500 SAY "Knoledge Base:"  ;
175:        FONT "Terminal", 8 ;
176:        STYLE "T"
177:
178:  IF NOT WVISIBLE("kbedit")
179:     ACTIVATE WINDOW kbedit
180:  ENDIF
181:
182:   READ CYCLE ;
183:      WHEN setaction()
184:
185:   RELEASE WINDOW kbedit
186:
187:   #REGION 0
188:
189:   SET readborder &rborder
190:
191:  IF m.talkstat = "ON"
192:     SET TALK ON
193:  ENDIF
194:  IF m.compstat = "ON"
195:     SET COMPATIBLE ON
196:  ENDIF
197:
198:
```

```
199:     *
200:     *
201:     *          KBEDIT/Windows Cleanup Code
202:     *
203:     *
204:     *
205:
206:     #REGION 1
207:     SELECT rule
208:     USE
209:     SELECT premise
210:     USE
211:     SELECT fact
212:     USE
213:     SELECT action
214:     USE
215:     SELECT (msel)
216: <---RETURN
217:
218:
219:
220:     *
221:     *
222:     *     KBEDIT/Windows Supporting Procedures and Functions
223:     *
224:     *
225:     *
226:
227:     #REGION 1
228:     PROCEDURE setprem
229:     PRIVATE msel, i, j, jj, K, N, p, R, s
230:     msel = SELECT()
231:     SELECT premise
232:     SEEK rule.premise
233:     i = 0
234:     N = 0
235:     prem = " "
236:
237:     * collect all triples
238:  DO WHILE clause = rule.premise .AND. !EOF()
239:        i = i + 1
240:        triple[i,1] = premise.op
241:        triple[i,2] = premise.fact
242:        triple[i,3] = premise.factr
243:        triple[i,4] = 0
244:        triple[i,5] = RECNO()
245:        SKIP
246:  ENDDO
247:     ntriples = i
248:
249:     * collect leaf nodes
250:     SELECT fact
251:  FOR j = 1 TO ntriples
252:        IF EMPTY( triple[j,1] )
253:           SEEK triple[j,2]
254:           IF FOUND()
255:              N = N + 1
256:              IF OBJECT = "S"
257:                 SELECT dict
258:              ELSE
259:                 SELECT disease
260:              ENDIF
261:              SEEK fact.id
262:              m.name = name
263:              SELECT fact
264:              m.val = TRIM( IIF( TAG = "T", LEFT( TEXT, 80), VAL ) )
```

```
265:                SELECT dict
266:
267:                FOR jj = 1 TO 2
268:                   K = AT ( "@", m.val, jj )
269:                   IF K = 0
270:                   EXIT
271:                   ENDIF
272:                   p = VAL( SUBSTR( m.val, K + 1 ) )
273:                   SEEK p
274:                   m.val = STRTRAN( m.val, "@" + LTRIM( STR( p ) ), "(" + TR
=> IM( name ) + ")"   )
275:                NEXT
276:
277:                SELECT fact
278:                prem[n] = TRIM( m.name ) + " " + op + " " + m.val
279:                premr[n] = j
280:                premo[n] = ""
281:                triple[j,4] = N
282:             ENDIF
283:          ENDIF
284:       NEXT
285:       SELECT premise
286:
287:       premm = .F.
288:       ptop = 0
289:       IF ntriples > 0
290:          =PUSH(ntriples)     && root of expression tree
291:       ENDIF
292:
293:       * parse the expression tree
294:       DO WHILE !sempty()
295:          i = POP()
296:          IF !EMPTY( triple[i, 1] )
297:             * operator node
298:             IF !premm[i]
299:                * not yet marked; defer
300:                premm[i] = .T.
301:                =PUSH( i )                    && push operator
302:                =PUSH( triple[i, 3] )         && ..right
303:                =PUSH( triple[i, 2] )         && ..left
304:             ELSE
305:                * already marked..
306:                j = triple[triple[i, 2], 4]  && left term
307:                K = triple[triple[i, 3], 4]  && right term
308:                triple[i, 4] = j             && right term
309:                IF LEFT( prem[k], 1 ) = triple[i,1]
310:                   * same operator; fold tree
311:                   s = hlink( SUBSTR( prem[j], 3 ) )  && add link
312:                   prem[j] = "┌─" + s                && add elbow
313:                   premo[k] = IIF(EMPTY(premo[k]),ALLTRIM(STR(i)),premo[k] +
=>  "^" + ALLTRIM(STR(i)))
314:                ELSE
315:                   s = hlink( prem[j] )
316:                   prem[j] = "┌─" + s          && complete left child
317:                   s = hlink(prem[k] )
318:                   prem[k] = triple[i,1] + "─" + s  && opcode and right chil
=> d
319:                   premo[k] = IIF(EMPTY(premo[k]),ALLTRIM(STR(i)),premo[k] +
=>  "^" + ALLTRIM(STR(i)))
320:                   * indent remaining terms
321:                   FOR R = 1 TO N
322:                      IF R <> j  .AND. R <> K
323:                         * indent
324:                         prem[r] = SPACE(2) + prem[r]
325:                      ENDIF
326:                   NEXT
327:                ENDIF
328:                * bridge missing links
329:                FOR R = 1 TO N
330:                   IF R > j .AND. R < K
331:                      s = prem[r]
332:                      IF EMPTY( LEFT( s, 1 ) )
333:                         prem[r] = "│" + SUBSTR( s, 2 )
334:                      ENDIF
335:                   ENDIF
336:                NEXT
337:                * cleanup
338:                FOR R = j TO K-1
339:                   * complete missing links
340:                   =vlink( "┌", "┌" )
341:                   =vlink( "└", "┌" )
342:                   =vlink( "│", "┌" )
343:                NEXT
344:             ENDIF
345:          ENDIF
346:       ENDDO
347:
348:       mpn = N
349:       mp  = MIN( mpn, mp )
350:       mp  = MAX( 1, mp )
351:       SELECT (msel)
352: <---  RETURN
353:
354:       FUNCTION vlink
355:       PARAMETERS FROM, TO
356:       PRIVATE i,s
357:       i = 1
358:       DO WHILE EMPTY( SUBSTR( prem[r], i, 1) )
359:          i = i + 1                  && find leading non-blank
360:       ENDDO
361:       s = prem[r+1]
362:       * complete missing links
363:       IF SUBSTR( prem[r], i, 1 ) = FROM .AND. SUBSTR( s, i, 1 ) = TO
364:          prem[r+1] = LEFT( s, i-1 ) + "├" + SUBSTR( s, i+1 )
365:       ENDIF
366: <---  RETURN ""
367:
368:       FUNCTION hlink
369:       PARAMETERS s
370:       PRIVATE p
371:       p = ""
372:       DO WHILE EMPTY( LEFT(s, 2 ) )
373:          p = p + "──"
374:          s = SUBSTR( s, 3 )
375:       ENDDO
376: <---  RETURN p + s
377:
378:       FUNCTION PUSH
379:       PARAMETERS N
380:       ptop = ptop + 1
381:       prems[ptop] = N
382: <---  RETURN N
383:
384:       FUNCTION POP
385:       PRIVATE N
386:       N = prems[ptop]
387:       ptop = ptop - 1
388: <---  RETURN N
389:
390:       FUNCTION sempty
391: <---  RETURN ptop < 1
392:
```

```
393:     PROCEDURE setact
394:     PRIVATE msel, i, j, K, p
395:     msel = SELECT()
396:     SELECT action
397:     SEEK rule.action
398:     i = 0
399:     act = " "
400:     actr = 0
401:    ┌DO WHILE clause = rule.action .AND. !EOF()
402:    │   i = i + 1
403:    │   actr[i] = RECNO()
404:    │  ┌IF OBJECT = "D"
405:    │  │   SELECT disease
406:    │  ├ELSE
407:    │  │   SELECT dict
408:    │  └ENDIF
409:    │   SEEK action.id
410:    │   m.name = name
411:    │   SELECT action
412:    │   m.val = TRIM( IIF( TAG = "T", LEFT( TEXT, 80), VAL ) )
413:    │   SELECT dict
414:    │  ┌FOR j = 1 TO 2
415:    │  │   K = AT ( "@", m.val, j )
416:    │  │ ┌IF K = 0
417:    │  v ├─────EXIT
418:    │  │ └ENDIF
419:    │  │   p = VAL( SUBSTR( m.val, K + 1 ) )
420:    │  │   SEEK p
421:    │  │   m.val = STRTRAN( m.val, "@" + LTRIM( STR( p ) ), "(" + TRIM( na
=> me ) + ")"   )
422:    │  └NEXT
423:    │   SELECT action
424:    │   act[i] = TRIM( m.name ) + " " + op + " " + m.val
425:    │   SKIP
426:    └ENDDO
427:     man = i
428:     ma = MIN(man, ma)
429:     ma = MAX(1, ma)
430:    ┌IF actr[ma] > 0
431:    │   GOTO actr[ma]
432:    └ENDIF
433:     SELECT (msel)
434: <──RETURN
435:
436:     PROCEDURE setelse
437:     PRIVATE msel, i, j, K, p
438:     msel = SELECT()
439:     SELECT action
440:     SEEK rule.else
441:     i = 0
442:     actelse = " "
443:     actelser = 0
444:    ┌DO WHILE clause = rule.else .AND. !EOF()
445:    │   i = i + 1
446:    │   actelser[i] = RECNO()
447:    │  ┌IF OBJECT = "D"
448:    │  │   SELECT disease
449:    │  ├ELSE
450:    │  │   SELECT dict
451:    │  └ENDIF
452:    │   SEEK action.id
453:    │   m.name = name
454:    │   SELECT action
455:    │   m.val = TRIM( IIF( TAG = "T", LEFT( TEXT, 80), VAL ) )
456:    │   SELECT dict
457:    │  ┌FOR j = 1 TO 2
458:    │  │   K = AT ( "@", m.val, j )
459:    │  │ ┌IF K = 0
460:    │  v ├─────EXIT
461:    │  │ └ENDIF
462:    │  │   p = VAL( SUBSTR( m.val, K + 1 ) )
463:    │  │   SEEK p
464:    │  │   m.val = STRTRAN( m.val, "@" + LTRIM( STR( p ) ), "(" + TRIM( na
=> me ) + ")"   )
465:    │  └NEXT
466:    │   SELECT action
467:    │   actelse[i] = TRIM( m.name ) + " " + op + " " + m.val
468:    │   SKIP
469:    └ENDDO
470:     man = i
471:    ┌IF EMPTY(man)
472:    │   actelse[1] = ""
473:    │   actelser[1] = 0
474:    ├ELSE
475:    │   me = MIN(man, me)
476:    │   me = MAX(1, me)
477:    │  ┌IF actelser[me] > 0
478:    │  │   GOTO actelser[me]
479:    │  └ENDIF
480:    └ENDIF
481:     SELECT (msel)
482: <──RETURN
483:
484:     FUNCTION setaction
485:    ┌IF EMPTY(prem[1])
486:    │   SHOW GET minvprem DISABLE
487:    │   SHOW GET mp DISABLE
488:    ├ELSE
489:    │   SHOW GET minvprem ENABLE
490:    │   SHOW GET mp ENABLE
491:    └ENDIF
492:    ┌IF EMPTY(act[1])
493:    │   SHOW GET minvact DISABLE
494:    │   SHOW GET ma DISABLE
495:    ├ELSE
496:    │   SHOW GET minvact ENABLE
497:    │   SHOW GET ma ENABLE
498:    └ENDIF
499:    ┌IF EMPTY(actelse[1])
500:    │   SHOW GET minvelse DISABLE
501:    │   SHOW GET me DISABLE
502:    ├ELSE
503:    │   SHOW GET minvelse ENABLE
504:    │   SHOW GET me ENABLE
505:    └ENDIF
506: <──RETURN
507:
508:     *
509:     *
510:     *
511:     *
512:     *
513:     *
514:     *
515:     *
516:     *
517:     *
518:     *
519:     *
520:     *
521:     *
522:     *
```

_QSFOKQOSA	minvprem VALID
Function Origin:	
From Platform:	Windows
From Screen:	KBEDIT, Record Number: 5
Variable:	minvprem
Called By:	VALID Clause
Object Type:	Push Button
Snippet Number:	1

```
523:    FUNCTION _qsf0kq0sa     &&  minvprem VALID
524:    #REGION 1
525:    PRIVATE msel
526:    msel = SELECT()
527:    SELECT premise
528:    LOCATE FOR clause = rule.premise
529:    SET TOPIC TO "PREMISE"
530:    DO term.spr
531:    DO setprem
532:    SELECT (msel)
533:    SHOW GETS
534:
535:    *
536:    *
537:    *         _QSF0KQ0WK            minvact VALID
538:    *
539:    *         Function Origin:
540:    *
541:    *         From Platform:        Windows
542:    *         From Screen:          KBEDIT,     Record Number:    7
543:    *         Variable:             minvact
544:    *         Called By:            VALID Clause
545:    *         Object Type:          Push Button
546:    *         Snippet Number:       2
547:    *
548:    *
549:    *
550:    FUNCTION _qsf0kq0wk     &&  minvact VALID
551:    #REGION 1
552:    PRIVATE msel
553:    msel = SELECT()
554:    SELECT action
555:    LOCATE FOR clause = rule.action
556:    SET TOPIC TO "ACTION"
557:    DO action.spr
558:    SELECT (msel)
559:    SHOW GETS
560:
561:    *
562:    *
563:    *         _QSF0KQ10R            minvelse VALID
564:    *
565:    *         Function Origin:
566:    *
567:    *         From Platform:        Windows
568:    *         From Screen:          KBEDIT,     Record Number:    9
569:    *         Variable:             minvelse
570:    *         Called By:            VALID Clause
571:    *         Object Type:          Push Button
572:    *         Snippet Number:       3
573:    *
574:    *
575:    *
576:    FUNCTION _qsf0kq10r     &&  minvelse VALID
577:    #REGION 1
578:    PRIVATE msel
579:    msel = SELECT()
580:    SELECT action
581:    LOCATE FOR clause = rule.else
582:    SET TOPIC TO "ACTION"
583:    DO actelse.spr
584:    DO setelse
585:    SHOW GETS
586:    SELECT (msel)
587:
588:    *
```

```
589:    *
590:    *         _QSF0KQ14Z            mbbutton VALID
591:    *
592:    *         Function Origin:
593:    *
594:    *         From Platform:        Windows
595:    *         From Screen:          KBEDIT,     Record Number:   11
596:    *         Variable:             mbbutton
597:    *         Called By:            VALID Clause
598:    *         Object Type:          Push Button
599:    *         Snippet Number:       4
600:    *
601:    *
602:    *
603:    FUNCTION _qsf0kq14z     &&  mbbutton VALID
604:    #REGION 1
605:    SELECT rule
606:    DO CASE
607:    CASE mbbutton = 1             && <Next>
608:       m.recno = RECNO()
609:       IF !EOF()
610:          SKIP
611:       ELSE
612:          GOTO BOTTOM
613:       ENDIF
614:       m.goback = area != m.areaid
615:       IF m.goback
616:          GOTO m.recno
617:       ENDIF
618:    CASE mbbutton = 2             && <previous>
619:       m.recno = RECNO()
620:       IF !BOF()
621:          SKIP -1
622:       ELSE
623:          GOTO TOP
624:       ENDIF
625:       m.goback  = area != m.areaid
626:       IF m.goback
627:          GOTO m.recno
628:       ENDIF
629:    CASE mbbutton = 3             && <Browse>
630:       SET TOPIC TO "BROWSE"
631:       BROWSE FIELDS rule,salience FOR area = m.areaid NOEDIT
632:    CASE mbbutton = 4             && <Edit>
633:       SET TOPIC TO "EDIT"
634:       DO rule.spr
635:    CASE mbbutton = 5             && <Quit>
636:       CLEAR READ
637:       RETURN
638:    ENDCASE
639:    SCATTER MEMVAR
640:    DO setprem
641:    DO setact
642:    DO setelse
643:    DO setaction
644:    SHOW GETS
645:    mtopic = ALIAS()
646:    SET TOPIC TO &mtopic
647: *:  EOF: KBEDIT.ac1
```

```
*
*       08/09/94              OBLIST.SPR                  09:40:10
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*

PARAMETERS marray, m.item

*
*            OBLIST/MS-DOS Setup Code - SECTION 1
*
*

#REGION 1
EXTERNAL ARRAY marray


#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

*
*              MS-DOS Window definitions
*
*

IF NOT WEXIST("w_obj") ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.PJX" ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.SCX" ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.MNX" ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.PRG" ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.FRX" ;
      OR UPPER(WTITLE("W_OBJ")) == "W_OBJ.QPR"
   DEFINE WINDOW w_obj ;
      FROM INT((SROW()-21)/2),INT((SCOL()-62)/2) ;
      TO INT((SROW()-21)/2)+20,INT((SCOL()-62)/2)+61 ;
      TITLE "Object list" ;
      NOFLOAT ;
      NOCLOSE ;
      SHADOW ;
      NOMINIMIZE ;
      COLOR SCHEME 1
ENDIF
```



```


*
*            OBLIST/MS-DOS Screen Layout
*
*

#REGION 1
IF WVISIBLE("w_obj")
   ACTIVATE WINDOW w_obj SAME
ELSE
   ACTIVATE WINDOW w_obj NOSHOW
ENDIF
@ 0,1 GET m.obj ;
   PICTURE "@&N" ;
   FROM marray ;
   SIZE 17,58 ;
   DEFAULT 1 ;
   COLOR SCHEME 2
@ 18,19 GET m.buttons ;
   PICTURE "@*HT \<New;\<Ok;\<Cancel" ;
   SIZE 1,8,1 ;
   DEFAULT 1 ;
   VALID _qsf0kq4m3()

IF NOT WVISIBLE("w_obj")
   ACTIVATE WINDOW w_obj
ENDIF

READ CYCLE

RELEASE WINDOW w_obj

#REGION 0
IF m.talkstat = "ON"
   SET TALK ON
ENDIF
IF m.compstat = "ON"
   SET COMPATIBLE ON
ENDIF


*
*            OBLIST/MS-DOS Cleanup Code
*
*

#REGION 1
*FUNCTION validf
*IF !seek(id,'quals')
*   RETURN right(str(id),5) + "  " + name
*ENDIF
*RETURN

*
*       _QSF0KQ4M3            m.buttons VALID
*
*       Function Origin:
*
*       From Platform:        MS-DOS
*       From Screen:          OBLIST,     Record Number:    3
```

```
133:     *          Variable:            m.buttons
134:     *          Called By:           VALID Clause
135:     *          Object Type:         Push Button
136:     *          Snippet Number:      1
137:     *
138:     *
139:     *
140:     FUNCTION _qsf0kq4m3     &&  m.buttons VALID
141:     #REGION 1
142:     DO CASE
143:     CASE m.buttons = 1
144:         m.item = ""
145:     CASE m.buttons = 2
146:         m.item = marray[m.obj]
147:     OTHERWISE
148:         m.item ="~"
149:     ENDCASE
150:  *:  EOF: OBLIST.ac1
```

```
*
*       08/09/94              DDEDIT.SPR                09:40:13
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*
PARAMETERS m.newid, m.newnm

*
*            DDEDIT/Windows Setup Code - SECTION 1
*
*

#REGION 1
PRIVATE mp, mpn, msel, m.adding
DIMENSION enum[20], enumr[20], mrules[1]
msel = SELECT()
SELECT dict
SET ORDER TO 1
enum = " "
enumr = 0
mrules = " "
mp = 1
m.adding = .F.
m.qrecno = RECNO()
IF PARAMETER() = 0
   m.newid = id
ENDIF

SEEK m.newid
IF !FOUND()
   SCATTER MEMVAR BLANK
   m.adding = .T.
   m.id = m.newid
   m.name = m.newnm
ELSE
   SCATTER MEMVAR
ENDIF
IF !m.adding
   DO setrules
ENDIF
DO setenum


#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET readborder ON

*
*            Windows Window definitions
*
*

IF NOT WEXIST("_qsf0kq5ys")
   DEFINE WINDOW _qsf0kq5ys ;
      AT  0.000, 0.000  ;
      SIZE 33.083,71.750 ;
      TITLE "Data Element Editor" ;
      FONT "Terminal", 8 ;
      FLOAT ;
      NOCLOSE ;
      MINIMIZE ;
      SYSTEM ;
      COLOR RGB(,,,128,128,128)
   MOVE WINDOW _qsf0kq5ys CENTER
ENDIF

*
*            DDEDIT/Windows Screen Layout
*
*

#REGION 1
IF WVISIBLE("_qsf0kq5ys")
   ACTIVATE WINDOW _qsf0kq5ys SAME
ELSE
   ACTIVATE WINDOW _qsf0kq5ys NOSHOW
ENDIF
@ 1.333,57.750 SAY "Type"  ;
   FONT "Terminal", 8
@ 29.167,12.000 SAY "Hi"  ;
   FONT "Terminal", 8
@ 29.167,31.500 SAY "Lo"  ;
   FONT "Terminal", 8
@ 29.333,49.500 SAY "Units"  ;
   FONT "Terminal", 8
@ 1.333,4.500 SAY "Id"  ;
   FONT "Terminal", 8
@ 1.417,13.500 SAY "Name"  ;
   FONT "Terminal", 8
@ 27.167,9.000 SAY "Width"  ;
   FONT "Terminal", 8
@ 27.167,37.500 SAY "Decimals"  ;
   FONT "Terminal", 8
@ 29.083,4.500 SAY "Range:"  ;
   FONT "Terminal", 8
@ 17.167,4.500 SAY "Enum"  ;
   FONT "Terminal", 8
@ 27.167,4.500 SAY "Val:"  ;
   FONT "Terminal", 8
@ 11.333,4.500 SAY "Question Askable:"  ;
   FONT "Terminal", 8 ;
   STYLE "T"
```

```
133:     @ 3.167,4.500 SAY "Use in rules:"  ;
134:        FONT "Terminal", 8 ;
135:        STYLE "T"
136:     @ 1.167,7.750 GET m.id ;
137:        SIZE 1.000,3.125 ;
138:        DEFAULT " " ;
139:        FONT "Terminal", 8 ;
140:        DISABLE
141:     @ 1.167,19.750 GET m.name ;
142:        SIZE 1.000,33.875 ;
143:        DEFAULT " " ;
144:        FONT "Terminal", 8
145:     @ 1.083,63.000 GET m.datatype ;
146:        PICTURE "@^ N;E;M;L" ;
147:        SIZE 1.500,4.375 ;
148:        DEFAULT "N" ;
149:        FONT "Terminal", 8 ;
150:        VALID _qsf0kq6m1() ;
151:        DISABLE
152:     @ 5.167,4.625 GET m.rulesuse ;
153:        PICTURE "@&N" ;
154:        FROM mrules ;
155:        SIZE 4.667,62.875 ;
156:        DEFAULT 1 ;
157:        FONT "Terminal", 8
158:     @ 11.333,22.750 GET m.askable ;
159:        SIZE 1.000,4.625 ;
160:        DEFAULT .F. ;
161:        FONT "Terminal", 8
162:     @ 13.167,4.750 EDIT m.question ;
163:        SIZE 3.000,62.875,0.000 ;
164:        DEFAULT " " ;
165:        FONT "Terminal", 8 ;
166:        SCROLL ;
167:        WHEN m.askable
168:     @ 19.167,4.625 GET mp ;
169:        PICTURE "@&N" ;
170:        FROM enum ;
171:        SIZE 7.000,62.875 ;
172:        DEFAULT 1 ;
173:        FONT "Terminal", 8 ;
174:        WHEN m.datatype $ "EML" ;
175:        VALID _qsf0kq6t0()
176:     @ 27.167,16.750 GET m.width ;
177:        SIZE 1.000,13.000 ;
178:        DEFAULT 0 ;
179:        FONT "Terminal", 8 ;
180:        WHEN m.datatype = "N" ;
181:        DISABLE
182:     @ 27.167,49.750 GET m.dec ;
183:        SIZE 1.000,16.000 ;
184:        DEFAULT 0 ;
185:        FONT "Terminal", 8 ;
186:        WHEN m.datatype = "N" ;
187:        DISABLE
188:     @ 29.167,16.750 GET m.hi ;
189:        SIZE 1.000,13.000 ;
190:        DEFAULT 0 ;
191:        FONT "Terminal", 8 ;
192:        WHEN m.datatype = "N" ;
193:        DISABLE
194:     @ 29.167,34.750 GET m.lo ;
195:        SIZE 1.000,13.000 ;
196:        DEFAULT 0 ;
197:        FONT "Terminal", 8 ;
198:        WHEN m.datatype = "N" ;
199:        DISABLE
200:     @ 29.333,55.750 GET m.units ;
201:        SIZE 1.000,10.000 ;
202:        DEFAULT " " ;
203:        FONT "Terminal", 8 ;
204:        WHEN m.datatype = "N" ;
205:        DISABLE
206:     @ 31.000,27.000 GET mbuttons ;
207:        PICTURE "@*HT OK;Cancel" ;
208:        SIZE 1.917,8.375,0.750 ;
209:        DEFAULT 1 ;
210:        FONT "Terminal", 8 ;
211:        VALID _qsf0kq6yy()
212:
213:     IF NOT WVISIBLE("_qsf0kq5ys")
214:        ACTIVE WINDOW _qsf0kq5ys
215:     ENDIF
216:
217:     READ CYCLE ;
218:        WHEN _qsf0kq74f()
219:
220:     RELEASE WINDOW _qsf0kq5ys
221:
222:     #REGION 0
223:
224:     SET readborder &rborder
225:
226:     IF m.talkstat = "ON"
227:        SET TALK ON
228:     ENDIF
229:     IF m.compstat = "ON"
230:        SET COMPATIBLE ON
231:     ENDIF
232:
233:
234:     *
235:     *
236:     *              DDEDIT/Windows Cleanup Code
237:     *
238:     *
239:     *
240:
241:     #REGION 1
242:     SELECT dict
243: <---RETURN
244:
245:     *-----------------------------------------------------------
246:     *
247:     * This procedure set the list of rules occupied the object
248:     *
249:     *-----------------------------------------------------------
250:
251:
252:     *
253:     *
254:     *        DDEDIT/Windows Supporting Procedures and Functions
255:     *
256:     *
257:     *
258:
259:     #REGION 1
260:     FUNCTION setrules
261:     PRIVATE msel,mvalid
262:     msel = SELECT()
263:     SELECT quals
264:     SET FILTER TO id = m.id AND OBJECT = "S"
```

```
265:      COUNT TO mcount
266:     IF mcount > 0
267:        DIMENSION mrules[mcount]
268:        i = 0
269:        SCAN
270:           i = i + 1
271:           mrules[i] = areaname(area) + "-- rules: " + ALLTRIM(STRTRAN(rul
=> es,"|",","))
272:           IF !EMPTY(ruleso)
273:              mrules[i] = mrules[i] + IIF(EMPTY(rules),"",",") ;
274:                 + ALLTRIM(STRTRAN(ruleso,"|",","))
275:           ENDIF
276:        ENDSCAN
277:     ELSE
278:        DIMENSION mrules[1]
279:        mrules[1]=""
280:     ENDIF
281:     SET FILTER TO
282:     SELECT (msel)
283: <==RETURN
284:
285:     PROCEDURE edenum
286:     SELECT enum
287:     IF enumr[mp] > 0
288:        GOTO enumr[mp]
289:     ENDIF
290:     *m.id = dict.id
291:     DO ddenum.spr
292:     DO setenum
293:     SHOW GETS
294:     SELECT dict
295: <==RETURN
296:
297:
298:     PROCEDURE setenum
299:     enum = " "
300:     enumr = 0
301:     IF m.datatype $ "EML"
302:        PRIVATE msel, i
303:        msel = SELECT()
304:        SELECT enum
305:        SEEK m.id
306:        i = 0
307:        DO WHILE id = m.id
308:           i = i + 1
309:           enumr[i] = RECNO()
310:           enum[i] = mutex + " " + TRIM( enumerate )
311:           SKIP
312:        ENDDO
313:        mpn = i
314:        mp = MIN(mpn, mp)
315:        mp = MAX(1, mp)
316:        IF enumr[mp] > 0
317:           GOTO enumr[mp]
318:        ENDIF
319:        SELECT (msel)
320:     ENDIF
321: <==RETURN
322:
323:     PROCEDURE setval
324:     PRIVATE m.dictid
325:     IF m.datatype $ "N"
326:        PRIVATE msel, i
327:        msel = SELECT()
328:        SELECT VAL
329:        m.dictid = m.id
330:        SEEK m.id
331:        IF FOUND()
332:           SCATTER MEMVAR
333:        ELSE
334:           SCATTER MEMVAR BLANK
335:           m.id = m.dictid
336:        ENDIF
337:        SHOW GETS
338:        SELECT (msel)
339:     ENDIF
340: <==RETURN
341:
342:     FUNCTION cleanenum
343:     PRIVATE msel
344:     msel = SELECT()
345:     SELECT enum
346:     SEEK m.id
347:     IF FOUND()
348:        DELETE FOR id = m.id
349:        PACK
350:     ENDIF
351:     SELECT (msel)
352: <==RETURN
353:
354:     FUNCTION areaname
355:     PARAMETER mid
356:     PRIVATE msel,mname
357:
358:     msel = SELECT()
359:     SELECT area
360:     SEEK mid
361:     IF FOUND()
362:        mname = name
363:     ELSE
364:        mname = ""
365:     ENDIF
366:     SELECT (msel)
367: <==RETURN mname
368:
369:
370:
371:     *
372:     *
373:     *        _QSF0KQ6M1            m.datatype VALID
374:     *
375:     *        Function Origin:
376:     *
377:     *        From Platform:        Windows
378:     *        From Screen:          DDEDIT,     Record Number:   17
379:     *        Variable:             m.datatype
380:     *        Called By:            VALID Clause
381:     *        Object Type:          Popup
382:     *        Snippet Number:       1
383:     *
384:     *
385:     *
386:     FUNCTION _qsf0kq6m1     &&  m.datatype VALID
387:     #REGION 1
388:     DO CASE
389:     CASE m.datatype $ "N"
390:        DO setval
391:        enum = " "
392:        SHOW GET mp DISABLE
393:        SHOW GET m.width ENABLE
394:        SHOW GET m.dec ENABLE
395:        SHOW GET m.hi ENABLE
```

```
396:        SHOW GET m.lo ENABLE
397:        SHOW GET m.units ENABLE
398:        SHOW GETS
399:     OTHERWISE
400:        DO setenum
401:        IF EMPTY(enum[1])
402:           DO ddenum.spr WITH .T.
403:           DO setenum
404:        ENDIF
405:        SHOW GET mp ENABLE
406:        SHOW GET m.width DISABLE
407:        SHOW GET m.dec DISABLE
408:        SHOW GET m.hi DISABLE
409:        SHOW GET m.lo DISABLE
410:        SHOW GET m.units DISABLE
411:        SHOW GETS
412:     ENDCASE
413:
414:     *
415:     *
416:     *        _QSF0KQ6T0            mp VALID
417:     *
418:     *        Function Origin:
419:     *
420:     *        From Platform:        Windows
421:     *        From Screen:          DDEDIT,     Record Number:   21
422:     *        Variable:             mp
423:     *        Called By:            VALID Clause
424:     *        Object Type:          List
425:     *        Snippet Number:       2
426:     *
427:     *
428:     *
429:     FUNCTION _qsf0kq6t0     &&  mp VALID
430:     #REGION 1
431:     DO edenum
432:
433:     *
434:     *
435:     *        _QSF0KQ6YY            mbuttons VALID
436:     *
437:     *        Function Origin:
438:     *
439:     *        From Platform:        Windows
440:     *        From Screen:          DDEDIT,     Record Number:   27
441:     *        Variable:             mbuttons
442:     *        Called By:            VALID Clause
443:     *        Object Type:          Push Button
444:     *        Snippet Number:       3
445:     *
446:     *
447:     *
448:     FUNCTION _qsf0kq6yy     &&  mbuttons VALID
449:     #REGION 1
450:     DO CASE
451:     CASE mbuttons = 1   && ok
452:        SELECT dict
453:        IF m.adding
454:           APPEND BLANK
455:        ENDIF
456:        GATHER MEMVAR
457:        IF m.datatype $ "N"
458:           = cleanenum()
459:           SELECT VAL
460:           IF m.adding
461:              APPEND BLANK
462:           ENDIF
463:           GATHER MEMVAR
464:        ENDIF
465:     CASE mbuttons = 2   && cancel
466:        IF m.adding
467:           = cleanenum()
468:        ENDIF
469:        mok = .F.
470:     ENDCASE
471:
472:
473:     *
474:     *
475:     *        _QSF0KQ74F            Read Level When
476:     *
477:     *        Function Origin:
478:     *
479:     *
480:     *        From Platform:        Windows
481:     *        From Screen:          DDEDIT
482:     *        Called By:            READ Statement
483:     *        Snippet Number:       4
484:     *
485:     *
486:     *
487:     FUNCTION _qsf0kq74f     && Read Level When
488:     *
489:     * When Code from screen: DDEDIT
490:     *
491:     #REGION 1
492:     IF m.adding
493:        SHOW GET m.datatype ENABLE
494:        SHOW GET m.rulesuse DISABLE
495:     ENDIF
496:     IF m.datatype $ "N"
497:        PRIVATE msel
498:        msel = SELECT()
499:        SELECT VAL
500:        SCATTER MEMVAR
501:        SHOW GET mp DISABLE
502:        SHOW GET m.width ENABLE
503:        SHOW GET m.dec ENABLE
504:        SHOW GET m.hi ENABLE
505:        SHOW GET m.lo ENABLE
506:        SHOW GET m.units ENABLE
507:        SHOW GETS
508:     ENDIF
509:  *:  EOF: DDEDIT.ac1
```

```
*
*         08/09/94            DDENUM.SPR              09:40:17
*
*
*         Author's Name
*
*         Copyright (c) 1994 Company Name
*         Address
*         City,     Zip
*
*         Description:
*         This program was automatically generated by GENSCRN.
*
*

PARAMETERS m.new

*
*
*         DDENUM/Windows Setup Code - SECTION 1
*
*
*

#REGION 1
m.dictid = m.id
IF PARAMETER() = 0
   m.enumadd = .F.
   SCATTER MEMVAR
ELSE
   m.dictid = m.id
   m.enumadd = .T.
   SELECT enum
   SEEK m.dictid
   IF !FOUND()
      m.enumadd = .T.
      m.ord = 1
      m.id = m.dictid
      m.mutex = "-"
   ELSE
      SCATTER MEMVAR
   ENDIF
ENDIF
SHOW GETS

#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
   SET TALK OFF
   m.talkstat = "ON"
ELSE
   m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET readborder ON

*
*
*         Windows Window definitions
*
*
*

IF NOT WEXIST("_qsf0kq96z")
   DEFINE WINDOW _qsf0kq96z ;
      AT 20.750, 7.500 ;
      SIZE 9.077,98.200 ;
      FONT "MS Sans Serif", 8 ;
      FLOAT ;
      NOCLOSE ;
      MINIMIZE ;
      SYSTEM
ENDIF


*
*
*         DDENUM/Windows Screen Layout
*
*
*

#REGION 1
IF WVISIBLE("_qsf0kq96z")
   ACTIVATE WINDOW _qsf0kq96z SAME
ELSE
   ACTIVATE WINDOW _qsf0kq96z NOSHOW
ENDIF
@ 5.154,5.000 SAY "Ord"  ;
   FONT "Terminal", 8
@ 1.462,4.800 SAY "Mutex"  ;
   FONT "Terminal", 8
@ 3.308,4.800 SAY "Enum"  ;
   FONT "Terminal", 8
@ 5.154,14.000 GET m.ord ;
   SIZE 1.000,2.000 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   DISABLE
@ 1.615,14.400 GET m.mutex ;
   SIZE 1.000,1.000 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   VALID _qsf0kq9mb()
@ 3.231,14.400 GET m.enumerate ;
   SIZE 1.000,47.750 ;
   DEFAULT " " ;
   FONT "Terminal", 8 ;
   PICTURE "@!"
@ 6.462,26.400 GET mbuttons ;
   PICTURE "@*HN Add;OK;Cancel" ;
   SIZE 1.917,7.500,0.750 ;
   DEFAULT 1 ;
   FONT "Terminal", 8 ;
   VALID _qsf0kq9qc()

IF NOT WVISIBLE("_qsf0kq96z")
   ACTIVATE WINDOW _qsf0kq96z
ENDIF

READ CYCLE

RELEASE WINDOW _qsf0kq96z

#REGION 0

```

```
133:     SET readborder &rborder
134:
135:    IF m.talkstat = "ON"
136:       SET TALK ON
137:    ENDIF
138:    IF m.compstat = "ON"
139:       SET COMPATIBLE ON
140:    ENDIF
141:
142:
143:     *
144:     *
145:     *          _QSF0KQ9MB            m.mutex VALID
146:     *
147:     *          Function Origin:
148:     *
149:     *          From Platform:        Windows
150:     *          From Screen:          DDENUM,      Record Number:    6
151:     *          Variable:             m.mutex
152:     *          Called By:            VALID Clause
153:     *          Object Type:          Field
154:     *          Snippet Number:       1
155:     *
156:     *
157:     *
158:     FUNCTION _qsf0kq9mb     &&  m.mutex VALID
159:     #REGION 1
160:    DO CASE
161:    CASE m.datatype $ "EL"
162: <------RETURN m.mutex = "-"
163:    CASE m.datatype $ "M"
164: <------RETURN m.mutex $ "-+"
165:    ENDCASE
166: <---RETURN .F.
167:
168:     *
169:     *
170:     *          _QSF0KQ9QC            mbuttons VALID
171:     *
172:     *          Function Origin:
173:     *
174:     *          From Platform:        Windows
175:     *          From Screen:          DDENUM,      Record Number:    8
176:     *          Variable:             mbuttons
177:     *          Called By:            VALID Clause
178:     *          Object Type:          Push Button
179:     *          Snippet Number:       2
180:     *
181:     *
182:     *
183:     FUNCTION _qsf0kq9qc     &&  mbuttons VALID
184:     #REGION 1
185:    DO CASE
186:    CASE mbuttons = 1   && add
187:       IF m.enumadd
188:          APPEND BLANK
189:          GATHER MEMVAR
190:       ENDIF
191:       m.enumadd = .T.
192:       m.neword = 0
193:       DO WHILE id = m.dictid
194:          m.neword = ord
195:          SKIP
196:       ENDDO
197:       SCATTER MEMVAR BLANK
198:       m.ord  =  m.neword + 1
199:       m.id  =  m.dictid
200:       m.mutex  =  "-"
201:       _CUROBJ = OBJNUM(m.mutex)
202:       SHOW GETS
203:    CASE mbuttons = 2   && ok
204:       IF m.enumadd
205:          APPEND BLANK
206:       ENDIF
207:       mok = .T.
208:       GATHER MEMVAR
209:       CLEAR READ
210:    CASE mbuttons = 3   && cancel
211:       mok = .F.
212:       CLEAR READ
213:    ENDCASE
214: *:  EOF: DDENUM.ac1
```

```
*
*       08/09/94          KBLOAD.SPR          09:40:19
*
*       Author's Name
*
*       Copyright (c) 1994 Company Name
*       Address
*       City,     Zip
*
*       Description:
*       This program was automatically generated by GENSCRN.
*
*
*
*
DO CASE
CASE _WINDOWS

   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS

   m.rborder = SET("READBORDER")
   SET readborder ON

   *       ┌──────────────────────────────────────────────────┐
   *       │                                                  │
   *       │              Windows Window definitions          │
   *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   IF NOT WEXIST("_qsf0kqb51")
      DEFINE WINDOW _qsf0kqb51 ;
         AT  0.000, 0.000  ;
         SIZE 22.333,45.500 ;
         TITLE "Knowledge Base Loader" ;
         FONT "Terminal", 8 ;
         FLOAT ;
         NOCLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         DOUBLE ;
         COLOR RGB(,,,128,128,128)
      MOVE WINDOW _qsf0kqb51 CENTER
   ENDIF

   *       ┌──────────────────────────────────────────────────┐
=> *       │                                                  │
   *       │          KBLOAD/Windows Setup Code - SECTION 2   │
=> *       │                                                  │
   *       │                                                  │
=> *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   m.home = CURDIR()
   m.drv   = m.new
   m.src   = ""
   m.fil   = ""
   m.ok = .F.
   SET DEFA TO (m.data)
   DEFINE POPUP listdir ;
      PROMPT FILES LIKE *. ;
      SCROLL ;
      MARGIN ;
      MARK " "
   ON ERROR m.src =""

   *       ┌──────────────────────────────────────────────────┐
=> *       │                                                  │
   *       │              KBLOAD/Windows Screen Layout        │
=> *       │                                                  │
   *       └──────────────────────────────────────────────────┘
   *

   #REGION 1
   IF WVISIBLE("_qsf0kqb51")
      ACTIVATE WINDOW _qsf0kqb51 SAME
   ELSE
      ACTIVATE WINDOW _qsf0kqb51 NOSHOW
   ENDIF
   @ 2.500,0.875 GET m.src ;
      SIZE 1.333,32.125 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      VALID _qsf0kqbnk()
   @ 7.583,1.375 GET m.new ;
      SIZE 1.167,31.750 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      VALID _qsf0kqbt0() ;
      DISABLE
   @ 9.917,1.000 GET m.dbf ;
      PICTURE "@&N" ;
      POPUP listdir ;
      SIZE 11.667,32.625 ;
      DEFAULT " " ;
      FONT "Terminal", 8 ;
      STYLE "" ;
      WHEN _qsf0kqc1j() ;
      VALID _qsf0kqc42()
   @ 14.250,34.875 GET m.load ;
      PICTURE "@*VN Load" ;
      SIZE 2.083,7.750,1.083 ;
      DEFAULT 1 ;
```

```
      FONT "Terminal", 8 ;
      WHEN loadok() ;
      VALID _qsf0kqc6t() ;
      DISABLE
   @ 17.417,34.875 GET mbuttonc ;
      PICTURE "@*VT Cancel" ;
      SIZE 2.250,7.875,1.083 ;
      DEFAULT 1 ;
      FONT "Terminal", 8
   @ 1.250,0.750 SAY "Read From Definition file:"  ;
      FONT "Terminal", 8
   @ 6.083,0.750 SAY "Create Files in Temporary directory:"  ;
      FONT "Terminal", 8

   IF NOT WVISIBLE("_qsf0kqb51")
      ACTIVATE WINDOW _qsf0kqb51
   ENDIF

   READ CYCLE MODAL

   RELEASE WINDOW _qsf0kqb51

   #REGION 0

   SET readborder &rborder

   IF m.talkstat = "ON"
      SET TALK ON
   ENDIF
   IF m.compstat = "ON"
      SET COMPATIBLE ON
   ENDIF


   *
   *
   *           KBLOAD/Windows Cleanup Code
   *
   *
   *

   #REGION 1
   SET DEFA TO (m.home)
   ON ERROR
   * load enable


CASE _DOS


   #REGION 0
   REGIONAL m.currarea, m.talkstat, m.compstat

   IF SET("TALK") = "ON"
      SET TALK OFF
      m.talkstat = "ON"
   ELSE
      m.talkstat = "OFF"
   ENDIF
   m.compstat = SET("COMPATIBLE")
   SET COMPATIBLE FOXPLUS


   *
   *
   *           MS-DOS Window definitions
   *
   *
   *

   IF NOT WEXIST("_qsf0kqci7")
      DEFINE WINDOW _qsf0kqci7 ;
         FROM INT((SROW()-18)/2),INT((SCOL()-55)/2) ;
         TO INT((SROW()-18)/2)+17,INT((SCOL()-55)/2)+54 ;
         TITLE "Knowledge Base Loader" ;
         FLOAT ;
         NOCLOSE ;
         SHADOW ;
         NOMINIMIZE ;
         DOUBLE ;
         COLOR SCHEME 5
   ENDIF


   *
   *
   *           KBLOAD/MS-DOS Setup Code - SECTION 2
   *
   *
   *

   #REGION 1
   m.home = CURDIR()
   m.drv   = m.new
   m.src   = ""
   m.fil   = ""
   m.ok = .F.
   SET DEFA TO (m.data)
   DEFINE POPUP listdir ;
      PROMPT FILES LIKE *. ;
      SCROLL ;
      MARGIN ;
      MARK " "
   ON ERROR m.src =""

   *
   *
   *           KBLOAD/MS-DOS Screen Layout
   *
   *
   *

   #REGION 1
```

```
IF WVISIBLE("_qsf0kqci7")
    ACTIVATE WINDOW _qsf0kqci7 SAME
ELSE
    ACTIVATE WINDOW _qsf0kqci7 NOSHOW
ENDIF
  @ 2,1 GET m.src ;
    SIZE 1,30 ;
    DEFAULT " " ;
    VALID _qsf0kqcp2()
  @ 6,1 GET m.new ;
    SIZE 1,30 ;
    DEFAULT " " ;
    VALID _qsf0kqcro() ;
    DISABLE
  @ 7,1 GET m.dbf ;
    PICTURE "@&N" ;
    POPUP listdir ;
    SIZE 9,30 ;
    DEFAULT " " ;
    WHEN _qsf0kqcua() ;
    VALID _qsf0kqcwt() ;
    COLOR SCHEME 6
  @ 13,42 GET m.load ;
    PICTURE "@*VT Load" ;
    SIZE 1,8,1 ;
    DEFAULT 1 ;
    WHEN loadok() ;
    VALID _qsf0kqczg() ;
    DISABLE
  @ 15,42 GET mbuttonc ;
    PICTURE "@*VT Cancel" ;
    SIZE 1,8,1 ;
    DEFAULT 1
  @ 1,1 SAY "Read From Definition file:" ;
    SIZE 1,26, 0
  @ 5,1 SAY "Create Files in Temporary directory:" ;
    SIZE 1,36, 0

IF NOT WVISIBLE("_qsf0kqci7")
    ACTIVATE WINDOW _qsf0kqci7
ENDIF

  READ CYCLE MODAL

  RELEASE WINDOW _qsf0kqci7

  #REGION 0
IF m.talkstat = "ON"
    SET TALK ON
ENDIF
IF m.compstat = "ON"
    SET COMPATIBLE ON
ENDIF


  *
  *
  *         KBLOAD/MS-DOS Cleanup Code
  *
  *
  *

    #REGION 1
    SET DEFA TO (m.home)
    ON ERROR
    * load enable

ENDCASE


*
*
*    _QSF0KQBNK          m.src VALID
*
*    Function Origin:
*
*    From Platform:      Windows
*    From Screen:        KBLOAD,     Record Number:    2
*    Variable:           m.src
*    Called By:          VALID Clause
*    Object Type:        Field
*    Snippet Number:     1
*
*
*
FUNCTION _qsf0kqbnk     &&  m.src VALID
#REGION 1
m.src = LOCFILE(m.src,"","Locate Definition File")
=loadok()


*
*
*    _QSF0KQBT0          m.new VALID
*
*    Function Origin:
*
*    From Platform:      Windows
*    From Screen:        KBLOAD,     Record Number:    3
*    Variable:           m.new
*    Called By:          VALID Clause
*    Object Type:        Field
*    Snippet Number:     2
*
*
*
FUNCTION _qsf0kqbt0     &&  m.new VALID
#REGION 1
=loadok()

*
*
*    _QSF0KQC1J          m.dbf WHEN
*
*    Function Origin:
*
*    From Platform:      Windows
*    From Screen:        KBLOAD,     Record Number:    4
*    Variable:           m.dbf
*    Called By:          WHEN Clause
*    Object Type:        List
*    Snippet Number:     3
*
*
*
FUNCTION _qsf0kqc1j     &&  m.dbf WHEN
```

```
#REGION 1
m.new = PRMBAR("listdir",2)
IF FILE(m.new + "\nul")
   SHOW OBJECT OBJNUM(m.new) ENABLE
ELSE
   SHOW OBJECT OBJNUM(m.new) DISABLE
ENDIF
=loadok()

*
*
*       _QSF0KQC42          m.dbf VALID
*
*       Function Origin:
*
*       From Platform:      Windows
*       From Screen:        KBLOAD,     Record Number:   4
*       Variable:           m.dbf
*       Called By:          VALID Clause
*       Object Type:        List
*       Snippet Number:     4
*
*
*
FUNCTION _qsf0kqc42     &&  m.dbf VALID
#REGION 1
=loadok()

*
*
*       _QSF0KQC6T          m.load VALID
*
*       Function Origin:
*
*       From Platform:      Windows
*       From Screen:        KBLOAD,     Record Number:   5
*       Variable:           m.load
*       Called By:          VALID Clause
*       Object Type:        Push Button
*       Snippet Number:     5
*
*
*
FUNCTION _qsf0kqc6t     &&  m.load VALID
#REGION 1
m.srcbak = m.src
m.valid = checkfile(m.new)
IF !m.valid
   SHOW GET m.load DISABLE
   _CUROBJ = OBJNUM(m.dbf)
   m.src = m.srcbak
   SHOW GETS
<--RETURN
ENDIF
olderr = ON("error")
ON ERROR RETURN
mrun = m.src + " " + m.new
m.temp = CURDIR()
m.t1 = LOCFILE("rulex","exe","where is file RULEX.EXE")
m.t1 = SUBSTR(m.t1, 1, AT("RULEX.EXE",m.t1) -1)
SET DEFA TO (m.t1)
RUN /400 rulex &mrun
SET DEFA TO (m.temp)
= popupshow("Working....")
DO kbldr WITH m.new
= popuphide()
ON ERROR &olderr
CLEAR READ


*
*
*       _QSF0KQCP2          m.src VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        KBLOAD,     Record Number:   11
*       Variable:           m.src
*       Called By:          VALID Clause
*       Object Type:        Field
*       Snippet Number:     6
*
*
*
FUNCTION _qsf0kqcp2     &&  m.src VALID
#REGION 1
m.src = LOCFILE(m.src)
=loadok()


*
*
*       _QSF0KQCRO          m.new VALID
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        KBLOAD,     Record Number:   12
*       Variable:           m.new
*       Called By:          VALID Clause
*       Object Type:        Field
*       Snippet Number:     7
*
*
*
FUNCTION _qsf0kqcro     &&  m.new VALID
#REGION 1
=loadok()

*
*
*       _QSF0KQCUA          m.dbf WHEN
*
*       Function Origin:
*
*       From Platform:      MS-DOS
*       From Screen:        KBLOAD,     Record Number:   13
*       Variable:           m.dbf
*       Called By:          WHEN Clause
*       Object Type:        List
*       Snippet Number:     8
*
*
*
FUNCTION _qsf0kqcua     &&  m.dbf WHEN
#REGION 1
m.new = PRMBAR("listdir",2)
IF FILE(m.new + "\nul")
   SHOW OBJECT OBJNUM(m.new) ENABLE
ELSE
   SHOW OBJECT OBJNUM(m.new) DISABLE
```

```
489:    └ENDIF
490:      =loadok()
491:
492:      *
493:      *
494:      *          _QSF0KQCWT            m.dbf VALID
495:      *
496:      *          Function Origin:
497:      *
498:      *          From Platform:        MS-DOS
499:      *          From Screen:          KBLOAD,     Record Number:   13
500:      *          Variable:             m.dbf
501:      *          Called By:            VALID Clause
502:      *          Object Type:          List
503:      *          Snippet Number:       9
504:      *
505:      *
506:      *
507:      FUNCTION _qsf0kqcwt     &&  m.dbf VALID
508:      #REGION 1
509:      =loadok()
510:
511:      *
512:      *
513:      *          _QSF0KQCZG            m.load VALID
514:      *
515:      *          Function Origin:
516:      *
517:      *          From Platform:        MS-DOS
518:      *          From Screen:          KBLOAD,     Record Number:   14
519:      *          Variable:             m.load
520:      *          Called By:            VALID Clause
521:      *          Object Type:          Push Button
522:      *          Snippet Number:       10
523:      *
524:      *
525:      *
526:      FUNCTION _qsf0kqczg     &&  m.load VALID
527:      #REGION 1
528:      mrun = m.src + " " + m.new
529:      m.temp = CURDIR()
530:      m.t1 = LOCFILE("rulex.exe")
531:      m.t1 = SUBSTR(m.t1, 1, AT("RULEX.EXE",m.t1) -1)
532:      SET DEFA TO (m.t1)
533:      RUN /400 rulex &mrun
534:      SET DEFA TO (m.temp)
535:      DO kbldr WITH m.new
536:
537:
538:
539:      *
540:      *          KBLOAD/MS-DOS Supporting Procedures and Functions
541:      *
542:      *
543:      *
544:      *
545:
546:      #REGION 1
547:
548:      *
549:      *
550:      *                    KBLOAD Function LOADOK
551:      *
552:      *
553:      *
554:
```

```
555:
556:      PROCEDURE loadok
557:     ┌DO CASE
558:     ├CASE _DOS
559:     │    m.ok = .T.
560:     │   ┌IF EMPTY(m.src)
561:     │   │    m.ok = .F.
562:     │   └ENDIF
563:     │    m.ok = m.ok .AND. FILE(m.src)
564:     │    m.new = ALLTRIM(m.new)
565:     │   ┌IF EMPTY(m.new)
566:     │   │    m.ok = .F.
567:     │   └ENDIF
568:     │   ┌IF RIGHT(m.new, 1) != "\"
569:     │   │    m.new = m.new + "\"
570:     │   └ENDIF
571:     │    m.d1 = IIF(RIGHT(m.data,1) != "\", m.data + "\", m.data)
572:     │   ┌IF UPPER(FULLPATH(m.new)) == UPPER(FULLPATH(m.d1))
573:     │   │    m.ok = .F.
574:     │   └ENDIF
575:     │   ┌IF !FILE(m.new + "nul")
576:     │   │    m.ok = .F.
577:     │   └ENDIF
578:     │   ┌IF m.ok
579:     │   │    SHOW OBJECT OBJNUM(m.load) ENABLE
580:     │   ├ELSE
581:     │   │    SHOW OBJECT OBJNUM(m.load) DISABLE
582:     │   └ENDIF
583: <═══════RETURN m.ok
584:     ├CASE _WINDOWS
585:     │    m.ok = .T.
586:     │   ┌IF EMPTY(m.src)
587:     │   │    m.ok = .F.
588:     │   └ENDIF
589:     │    m.ok = m.ok .AND. FILE(m.src)
590:     │    m.new = ALLTRIM(m.new)
591:     │   ┌IF EMPTY(m.new)
592:     │   │    m.ok = .F.
593:     │   └ENDIF
594:     │   ┌IF RIGHT(m.new, 1) != "\"
595:     │   │    m.new = m.new + "\"
596:     │   └ENDIF
597:     │    m.d1 = IIF(RIGHT(m.data,1) != "\", m.data + "\", m.data)
598:     │   ┌IF UPPER(FULLPATH(m.new)) == UPPER(FULLPATH(m.d1))
599:     │   │    m.ok = .F.
600:     │   └ENDIF
601:     │   ┌IF !FILE(m.new + "nul")
602:     │   │    m.ok = .F.
603:     │   └ENDIF
604:     │   ┌IF m.ok
605:     │   │    SHOW OBJECT OBJNUM(m.load) ENABLE
606:     │   ├ELSE
607:     │   │    SHOW OBJECT OBJNUM(m.load) DISABLE
608:     │   └ENDIF
609: <═══════RETURN m.ok
610:     │
611:     └ENDCASE
612:
613:      *
614:      *
615:      *                    KBLOAD Function CHECKFILE
616:      *
617:      *
618:      *
619:
620:      FUNCTION checkfile
```

```
621:     PARAMETER m.new
622:     PRIVATE m.exist, m.fload
623:     m.exist = .F.
624:    ┌DO CASE
625:    ├CASE FILE(m.new + "area.dbf")
626:    │    m.exist = .T.
627:    ├CASE FILE(m.new + "rule.dbf")
628:    │    m.exist = .T.
629:    ├CASE FILE(m.new + "premise.dbf")
630:    │    m.exist = .T.
631:    ├CASE FILE(m.new + "fact.dbf")
632:    │    m.exist = .T.
633:    ├CASE FILE(m.new + "action.dbf")
634:    │    m.exist = .T.
635:    ├CASE FILE(m.new + "disease.dbf")
636:    │    m.exist = .T.
637:    ├CASE FILE(m.new + "dict.dbf")
638:    │    m.exist = .T.
639:    ├CASE FILE(m.new + "val.dbf")
640:    │    m.exist = .T.
641:    ├CASE FILE(m.new + "enum.dbf")
642:    │    m.exist = .T.
643:    └ENDCASE
644:     m.fload = .T.
645:    ┌IF m.exist
646:    │    m.fload = yesno("Database in " + m.new + " already exist, overwrit
=> e & delete it?","YES","NO")
647:    └ENDIF
648: <──RETURN m.fload
649:
650:  *:  EOF: KBLOAD.ac1
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	JUNE 1996	FINAL JAN 1994-1995

4. TITLE AND SUBTITLE
TECHNICAL MANUAL FOR THE NAVY COMPUTER ASSISTED MEDICAL DIAGNOSIS KNOWLEDGE BASE EDITOR (NCAMD-KBE), VERSION 1.0

5. FUNDING NUMBERS
Program Element: 63706N
Work Unit Number:
M0095.005-6103

6. AUTHOR(S)
HOA L. LY AND DIANNA M. PEARSALL

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Health Research Center
P. O. Box 85122
San Diego, CA 92186-5122

8. PERFORMING ORGANIZATION
TECHNICAL DOCUMENT 96-4D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Medical Research and Development Command
National Naval Medical Center
Building 1, Tower 2
Bethesda, MD 20889-5044

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This technical manual contains the information on the program source code, data elements, and the database structure needed to maintain the Navy Computer Assisted Medical Diagnosis Knowledge Base Editor (NCAMD-KBE). This documentation was created using the FOXDOC version 2.5a program.

14. SUBJECT TERMS
Computer diagnosis
Technical Manual

15. NUMBER OF PAGES
181

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited